Desarrollo de un sistema web de comunicación para gasolineras mediante WebSockets en tiempo real en Gasintec S.A.

Pérez Jerez, Juan José

Departamento de Ciencias de la Computación

Carrera de Ingeniería de Sistemas e Informática

Trabajo de titulación, previo a la obtención del título de Ingeniero en Sistemas e Informática

Ing. Galárraga Hurtado, Juan Fernando

24 de agosto de 2022

TesisFinal_PEREZ.pdf

Scanned on: 2:11 August 18, 2022 UTC

Identical Words	160
Words with Minor Changes	142
Paraphrased Words	2
Omitted Words	969

JUAN FERNANDO GALARRAGA HURTADO

Firmado digitalmente por JUAN FERNANDO GALARRAGA HURTADO
Fecha: 2022.08.17 21:40:01 -05'00'

s

Departamento de Ciencias de la Computación

Carrera de Ingeniería de Sistemas e Informática

Certificación

Certifico que el trabajo de titulación: "**Desarrollo de un sistema web de comunicación para gasolineras mediante WebSockets en tiempo real en Gasintec S.A.**" fue realizado por el señor **Pérez Jerez, Juan José;** el mismo que cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, además fue revisado y analizado en su totalidad por la herramienta de prevención y/o verificación de similitud de contenidos; razón por la cual me permito acreditar y autorizar para que se lo sustente públicamente.

Sangolquí, 24 de agosto de 2022

JUAN FERNANDO GALARRAGA HURTADO

Ing. Galárraga Hurtado, Juan Fernando

C. C. 1711464816

Departamento de Ciencias de la Computación

Carrera de Ingenieria de Sistemas e Informática

Responsabilidad de Autoria

Yo, **Pérez Jerez, Juan José**, con cédula de ciudadanía N°1804474466, declaro que el contenido, ideas y criterios del trabajo de titulación: **"Desarrollo de un sistema web de comunicación para gasolineras mediante WebSockets en tiempo real en Gasintec S.A."** es de mi autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos, y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Sangolquí, 24 de agosto de 2022

.....

Pérez Jerez Juan José

C.C.: 1804474466

Departamento de Ciencias de la Computación

Carrera de Ingeniería de Sistemas e Informática

Autorización de Publicación

Yo **Pérez Jerez, Juan José**, con cédula de ciudadanía n°1804474466, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: **Titulo: Desarrollo de un sistema web de comunicación para gasolineras mediante WebSockets en tiempo real en Gasintec S.A.** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi responsabilidad.

Sangolquí, 24 de agosto de 2022

Pérez Jerez, Juan José

C.C.: 1804474466

Dedicatoria

Este pequeño logro lo quiero dedicar a mi madre, gracias por su paciencia, por su bondad y su gran apoyo en mi formación como ser humano y como universitario, sin sus palabras nada de lo que soy ahora sería posible.

Agradecimiento

Agradezco primero a Dios por la vida, y por tenerte aquí en este momento, a mis padres María Jerez y Gonzalo Pérez, son un ejemplo de lucha y constancia, en los momentos donde me daba por vencido y botaba la toalla fueron quienes me dieron palabras sabias para continuar. A mis amigos y hermanos que siempre confiaron en mí, y entre bromas supieron meterme presión para lograr este objetivo. A mis abuelitos Manuel, Sofía y Luz María que siempre me cuidan desde el cielo. Al ingeniero Fernando Galárraga por ser un guía en los últimos niveles de la carrera universitaria y en el presente trabajo.

Gracias.

Índice

Certificado del director 3

Autoría de responsabilidad 4

Autorización 5

Dedicatoria 6

Agradecimiento 7

Índice 8

Índice de tablas 12

Índice de figuras 13

Resumen 14

Abstract 15

Capítulo I: Introducción 16

- Antecedentes 16
- Problemática 17
- Justificación 18
- Objetivos 19
  - Objetivo General 19
  - Objetivos Específicos 19
- Alcance 19

Capítulo II: Marco Referencial 22
Metodología SCRUM 22
Ciclo de vida 22
Componentes 24
Elementos 26
WebSockets 30
Tiempo Real 31
Trabajos Relacionados 34
Capítulo III: Especificación de Requerimientos. 39
Propósito 39
Alcance 39
Definiciones, Acrónimos y Abreviaturas. 40
Definiciones 40
Abreviaturas 42
Referencias 42
Perspectivas del Producto 42
Funciones del Producto 43
Características de los usuarios 44
Restricciones 44
Interfaces de Software 45

Historias de Usuario 46

Requerimiento Funcionales 49

Requerimientos No Funcionales 50

Capítulo IV: Desarrollo e Integración. 52

Levantamiento de Requerimientos 52

Product Backlog 56

Historias de Usuario 56

Sprint Planning 60

Sprint Backlog 0 61

Sprint Review 0 62

Entregables Sprint 0 63

Diagrama Macro 64

Vista Principal de la Aplicación 66

Framework de Desarrollo 67

Sprint Backlog 1 67

Sprint Review 1 69

Entregables Sprint 1 69

Login 70

Visualización de lados de los surtidores 72

Información en tiempo real de los lados de los surtidores 73

Sprint Backlog 2 89

Sprint Review 2 91

Entregables Sprint 2 92

Visualización de la información 93

Búsqueda de clientes registrados 94

Búsqueda de clientes nuevos y registro de placas 96

Capítulo V: Deployment 100

Configuración 100

Verificación 101

Capítulo VI: Conclusiones y Recomendaciones 104

Conclusiones 104

Recomendaciones 104

Bibliografía 106

Índice de tablas

Tabla 1 Historia de Usuario 28
Tabla 2 Prioridades de estimación 29
Tabla 3 Sprint Backlog 30
Tabla 4 Historia de Usuario 1 46
Tabla 5 Historia de Usuario 2 46
Tabla 6 Historia de Usuario 3 47
Tabla 7 Historia de Usuario 4 48
Tabla 8 Historia de Usuario 5 48
Tabla 9 Historia de Usuario 6 49
Tabla 10 Requerimientos Funcionales 50
Tabla 11 Requerimientos No Funcionales 51
Tabla 12 Roles de equipo 52
Tabla 13 Formas de Pago 55
Tabla 14 Product Backlog - Historia de Usuario 1 56
Tabla 15 Product Backlog - Historia de Usuario 2 57
Tabla 16 Product Backlog - Historia de Usuario 3 58
Tabla 17 Product Backlog - Historia de Usuario 4 59
Tabla 18 Product Backlog - Historia de Usuario 5 59
Tabla 19 Product Backlog - Historia de Usuario 6 60
Tabla 20 Sprint Planning 61
Tabla 21 Sprint Backlog 0 62
Tabla 22 Sprint Review 0 63
Tabla 23 Sprint Backlog 1 68
Tabla 24 Sprint Review 1 69
Tabla 25 Características del software en tiempo real aplicado a Poll con consulta a la base de datos 76
Tabla 26 Características del software en tiempo real aplicado a Poll con websocket 78
Tabla 27 Características del software en tiempo real aplicado a JavaScript con websocket 81
Tabla 28 Características del software en tiempo real aplicado a JavaScript con websocket y API Fetch 85
Tabla 29 Asignación de colores a estado 87
Tabla 30 Sprint Backlog 2 90
Tabla 31 Sprint Review 2 92
Tabla 32 Agrupación de dígitos 97
Tabla 33 Multiplicación de valores impares 97
Tabla 34 Sumatoria de dígitos 97

Índice de figuras

Figura 1 Problemática administrativa en las estaciones de servicio. 18
Figura 2 Pilares Fundamentales 24
Figura 3 Elementos de Scrum 26
Figura 4 Diagrama de funcionamiento 64
Figura 5 Diagrama de acciones 65
Figura 6 Vista Principal de la aplicación 66
Figura 7 Vista de Login 70
Figura 8 Usuario no registrado 71
Figura 9 Usuario y/o clave incorrecta 71
Figura 10 Vista de los lados de los surtidores 72
Figura 11 Información del lado 73
Figura 12 Llamado a la función transaccionar mediante el componente Poll 75
Figura 13 Llamado a la función consultarWebsocket() mediante el componente Poll 77
Figura 14 Utilización de librerías para abrir una conexión websocket 79
Figura 15 Ejemplo de creación del objeto websocket 79
Figura 16 Vista de los lados con información 80
Figura 17 Vista de los lados sin colores 88
Figura 18 Vista de los lados con colores 89
Figura 19 Vista de Registro de Clientes sin información 93
Figura 20 Vista de registro de clientes cuando existe información en la base de datos 94
Figura 21 Búsqueda de un cliente registrado en la base de datos 95
Figura 22 Selección de cliente registrado en la base de datos 95
Figura 23 Vista del lado con información de cliente registrado 96
Figura 24 Registro de cliente nuevo 98
Figura 25 Vista de registro de placas 99
Figura 26 Mensaje de error cuando la placa no coincide con ningún patrón de validación 99
Figura 27 Ruta de librería en el servidor Wildfly 100
Figura 28 Archivo module.xml 100
Figura 29 Comando de configuración JVM 101
Figura 30 Archivos de deployment 101
Figura 31 Aplicación no desplegada 102
Figura 32 Aplicación desplegada 103

Resumen

El presente trabajo de investigación realiza un estudio del software en tiempo real con la utilización del protocolo websocket en un sistema desarrollado para el control de ventas en una estación de servicio. Para determinar la mejor herramienta se realizó una comparación de distintas tecnologías como lo son Poll, un componente Ajax de Primefaces, y WebSockets en la interacción de los lenguajes Java y JavaScript. Para el manejo de Poll se requiere intervalos definidos para su ejecución con el limitante de la latencia al presentar la información al usuario y que la información se presenta bajo petición. El protocolo websocket es más seguro y no existe limitación alguna por el tiempo, la latencia que existe es aceptable, gracias a la API Fetch de JavaScript que proporciona una forma simple y lógica de recuperar recursos de manera asíncrona a través de la red, se logró un control del manejo de memoria, no se logró limitar el uso de memoria RAM del navegador cuando se usa la aplicación, pero el API la controla en valores manejables. Para el ingreso de información existen validadores según las normativas de los entes de control. Considerar que los resultados presentados son ejecutados en un ambiente de pruebas, con un simulador de surtidor de gasolina.

Palabras clave: tiempo real, WebSocket, Poll, estación de servicio.

Abstract

The present research work carries out a study of the software in real time with the use of the websocket protocol in a system developed for the control of sales in a service station. To determine the best tool, a comparison of different technologies was made, such as Poll, an Ajax component of Primefaces, and WebSockets in the interaction of Java and JavaScript languages. For the management of Poll, defined intervals are required for its execution with the limitation of latency when presenting the information to the user and that the information is presented on request. The websocket protocol is more secure and there is no time limitation, the latency that exists is acceptable, thanks to the JavaScript Fetch API that provides a simple and logical way to retrieve resources asynchronously through the network, it was achieved a control of the memory management, it was not possible to limit the use of RAM memory of the browser when using the application, but the API controls it in manageable values. For the entry of information there are validators according to the regulations of the control entities. Consider that the results presented are executed in a test environment, with a gasoline pump simulator.

Key words: real time, WebSocket, Poll, fuel station.

Capítulo I: Introducción

Antecedentes

El sector petrolero del Ecuador es una actividad importante del sector y este hidrocarburo se conoce como petróleo, crudo u oro negro y los lugares donde se encuentra se denominan yacimientos. Los primeros yacimientos fueron descubiertos por Angulo en Ancón de Santa Elena (Comercio, 2012).

Una estación de servicio es el establecimiento que se dedica a la venta de gasolinas y diésel al público en general, estos son suministrados de sus depósitos a los tanques de los vehículos (Evequoz, 2005). En cada estación existen uno o varios surtidores, los cuales se encargan de despachar el combustible a los automotores a través de una pistola. Los derivados de petróleo que se comercializan en una estación de servicio con frecuencia en el Ecuador son Diésel, Extra, Eco País y Súper.

El proceso de venta de combustible empieza con la compra a la comercializadora, Petróleos y Servicios, Primax, Terpel, etc., este se transporta en autotanques hasta la estación de servicio donde será almacenado en tanques. Para despachar lo harán mediante surtidores, en el mercado se los tiene uno, dos o tres productos (Cuenca Navarrete, 2019). El vehículo se ubica en el surtidor que tenga el producto deseado y mediante una manguera es suministrado.

Gasintec S.A. es una empresa que “provee soluciones innovadoras, encaminadas a satisfacer y solventar las necesidades empresariales del sector industrial, a través de la implementación de aplicaciones innovadoras, utilizando tecnología de avanzada, líderes en el mercado nacional” (Gasintec, 2020).

La administración de una estación de servicio conlleva varias acciones y controles para que se realice correctamente. Uno de los más importantes es controlar el valor vendido y designar correctamente

el cliente despachado, para este control Gasintec actualmente tienen una versión de escritorio arrendada, los costos de mantenimiento y actualización son altos.

Problemática

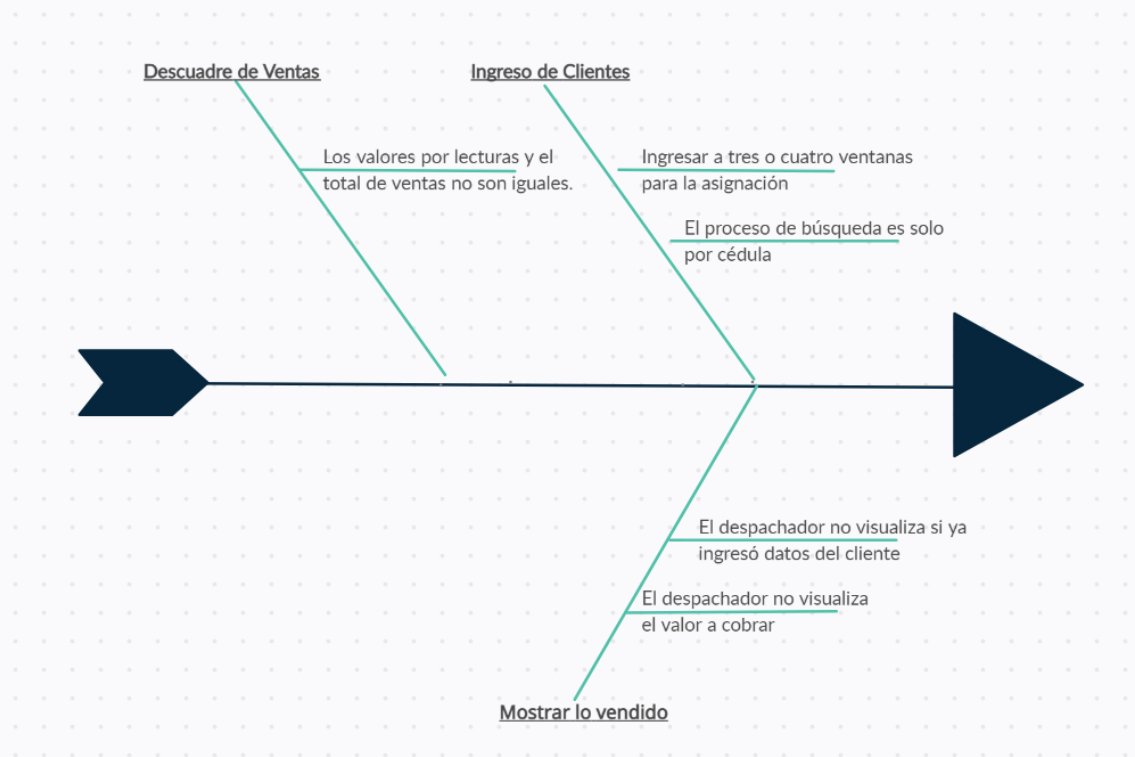
Las estaciones de servicio se manejan por turnos, al iniciar cada turno el despachador anota la lectura inicial de cada manguera, la lectura puede ser mecánica o digital, y al finalizar anota la lectura con la que se retira, con esto se realiza el cálculo de lo vendido, este valor de sebe comparar con las ventas realizadas a los clientes. Las ventas se calculan con lo vendido en efectivo y los créditos. Lo normal es que entre lecturas y ventas exista una diferencia mínima, en GASINTEC manifiestan que esta no debería pasar de un dólar, pero si existen diferencias mayores y al no tener un control correcto de despacho la verificación de las causas se vuelve complicada.

Para el ingreso de la información de los clientes el despachador lo busca por cédula, teniendo el inconveniente de que el cliente no le proporcione la información, e ingresa cualquiera o de la de él, esto conlleva a que puedan tener problemas legales con el S.R.I tanto el cliente/despachador como el propietario de la estación de servicio.

Los trabajadores de las estaciones de servicio, más conocidos como despachadores, tienen una instrucción académica baja o a su vez son personas de edad avanzada y el manejo de la tecnología se les dificulta, esto implica la demora en aprender el uso de un sistema, teniendo como consecuencia una demora en la atención al cliente.

Figura 1

Problemática administrativa en las estaciones de servicio.



Un sistema web que muestre la información de los surtidores, muestre lo vendido en tiempo real y facilite el ingreso/actualización de la información de los clientes facilitará la administración de una estación de servicio, tenga un control efectivo sobre lo facturado y con ello proporcionar al propietario de la gasolinera información relevante para un mejor manejo gerencial.

Justificación

Una estación de servicio se compone de varios aspectos para su correcto funcionamiento: eléctrica, hidráulica, ambiental, contable y de facturación. En este proyecto nos centraremos en el control de la facturación. Los surtidores de gasolina son los que emiten la información de lo vendido, tanto en

volumen como en dinero. La mayoría de propietarios de gasolineras se quejan por el tiempo de venta, la demora al ingresar información del cliente o las alternativas de ingreso.

El encargado del ingreso de información y cobro es el despachador, mientras más facilidades se le conceda en el software, relativamente menos errores cometerá y tendrá un despacho más ágil.

La empresa Gasintec S.A. justifica la necesidad de realizar un sistema web de comunicación para gasolineras mediante WebSockets en tiempo real con el objetivo de tener un mejor control del proceso de venta de combustibles, proporcionando al usuario(despachador) una herramienta amigable e intuitiva que mejore el tiempo de venta.

Objetivos

Objetivo General

Desarrollar un sistema web con comunicación WebSocket en tiempo real para el despacho de combustible en Estaciones de Servicio en Gasintec S.A.

Objetivos Específicos

- Realizar una revisión bibliográfica de la tecnología a utilizar para emplearlo en el desarrollo del sistema.
- Analizar y desarrollar los módulos de conexión WebSocket e ingreso de información de clientes.
- Verificar el funcionamiento del sistema en tiempo real y análisis de resultados.

Alcance

El desarrollo de un sistema web de comunicación para gasolineras mediante WebSockets en tiempo real en Gasintec S.A. será realizada con la metodología SCRUM, y tendrá el siguiente proceso:

- Realizar una revisión bibliográfica de la tecnología a utilizar para emplearlo en el desarrollo del sistema.

- Análisis de WebSockets.
- Análisis de Software en Tiempo Real.
- Estudios previos relacionados.

- Analizar y desarrollar los módulos de conexión WebSocket e ingreso de información de clientes.
  - Se mostrarán los lados de cada surtidor de despacho que tiene la gasolinera
  - Cada lado tendrá el valor de la venta en efectivo, el volumen despachado, el precio unitario del artículo y la verificación si ya ingresó datos para la venta.
  - Conectar el sistema web con el sistema de control de los surtidores que proporcionará Gasintec S.A.
  - Al presionar en un lado se mostrará una ventana nueva con la siguiente información:
    - Código
    - Identificación
    - Nombre
    - Placa
    - Lista de Placas
    - Forma de Pago
    - Tipo de Venta
  - Se buscará la información de un cliente nuevo por placa e identificación, el web service de búsqueda lo proporcionará Gasintec S.A.

  - Se buscará la información de un cliente antiguo por código, placa, identificación y nombre.
  - Una vez ingresada la información esta no podrá ser borrada, solo se puede actualizar.

- Verificar el funcionamiento del sistema en tiempo real y análisis de resultados.
  - Verificar que el sistema cumpla con las características del tiempo real.

Capítulo II: Marco Referencial

En el presente capítulo se describe la metodología y las herramientas utilizadas para alcanzar el resultado deseado en el tiempo estimado.

Metodología SCRUM

El desarrollo de software tiene sus dificultades, por ello existe una gran cantidad de metodologías para que su proceso sea llevado de una mejor manera. Existen propuestas que se centran en el proceso en sí, y otras que lo hacen en el factor humano y el producto software, es aquí donde entran las metodologías ágiles y sus principios.

Scrum es un proceso ágil y ligero para gestionar y controlar el desarrollo de software. El desarrollo es iterativo e incremental (una iteración es un ciclo de construcción breve y repetido). Cada ciclo o iteración finaliza con un software ejecutable que contiene una nueva funcionalidad. Las iteraciones suelen durar de 2 a 4 semanas. Scrum se utiliza como marco para otras prácticas de desarrollo de software como RUP y Extreme Programming. (MARTEL, 2015).

"The New Product Developroent Game" es un artículo publicado por Takeuchi y Nonaka en 1986. En él, presentan una nueva forma de gestionar proyectos cuyos elementos clave son la agilidad, la flexibilidad y la incertidumbre. Cuando observaron las empresas de tecnología, encontraron que se empleaban equipos multidisciplinarios para lograr el producto final. Esta forma de trabajo en equipo se ha comparado con los jugadores de rugby que trabajan juntos y usan una formación llamada SCRUM (Trigás Gallego, 2012).

Scrum es adecuado para proyectos de desarrollo que se caracterizan por tener entornos con incertidumbre, auto organización, control moderado y transmisión del conocimiento.

Ciclo de vida

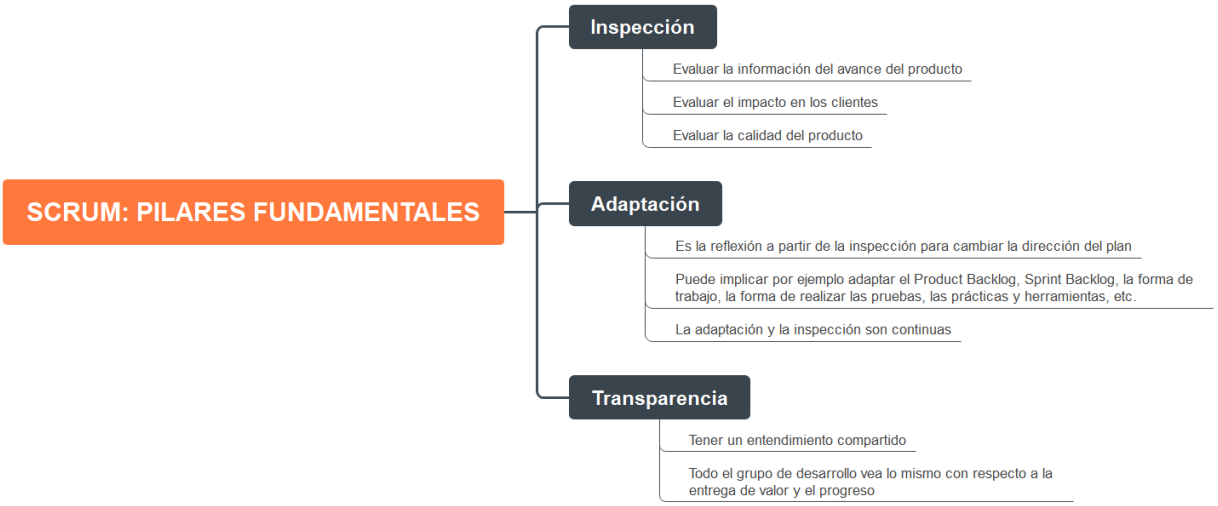
El ciclo de desarrollo Scrum se basa en cinco fases definidas como el ciclo de desarrollo ágil.

- **Concepto:** Característica del producto y asignada a un equipo de desarrollo.
- **Especulación:** Con la información obtenida se establecen límites de la iteración, costes y agendas, esta fase se repite y consiste en tres puntos: desarrollar y revisar los requisitos, mantener la lista de funcionalidades y un plan de entrega.
- **Exploración:** Se añaden las funcionalidades de especulación y se incrementa el producto.
- **Revisión:** El equipo de desarrollo revisa todas las funcionalidades basándose en el objetivo deseado.
- **Cierre:** Es la entrega de la primera versión funcional del producto, esto no significa que el proyecto esté completo. Los cambios, llamados mantenimiento, continúan hasta que se obtiene el producto final deseado.

Existen 3 pilares fundamentales para un correcto desarrollo del proyecto y de cada iteración: inspección, adaptación y transparencia como lo dice la figura 2.

Figura 2

Pilares Fundamentales.



Componentes

El proceso de desarrollo de tiene como componentes las reuniones y roles, las reuniones a su vez en 3 fases generales.

Las Reuniones

1. *Planificación del Backlog*

Se crea una lista de requisitos del sistema en orden de prioridad. En esta fase también se realiza la planificación del Sprint 0. Se determina cuál es el objetivo, qué trabajo debe realizarse en cada iteración y se realiza una lista de tareas siendo el objetivo más importante del sprint.

2. *Seguimiento del Sprint*

Son reuniones diarias para evaluar el avance de las tareas y se definirá por 3 preguntas, ¿Qué trabajo se realizó la reunión anterior?, ¿Qué trabajo se hará hasta una nueva reunión?, Inconvenientes que han surgido y que hay que solucionar para poder continuar.

Son reuniones diarias para evaluar el progreso de las tareas y están definidas por tres preguntas: ¿Qué trabajo se realizó en la última reunión?, ¿Qué trabajo se está realizando en el período previo a la nueva reunión?, y si ha surgido un inconveniente que debe resolverse para brindar más asistencia.

3. Revisión del Sprint

Al final del sprint, el Incremento generado se revisa para presentarse o crear una demostración para mostrar al cliente.

Roles

Cerdos

Son las personas que interfieren en todo el proceso del proyecto.

- **Product Owner:** Se encarga de la toma de decisiones, conoce el negocio del cliente, ordena prioritariamente los requerimientos y desarrolla el Product Backlog.
- **ScrumMaster:** Comprueba que la metodología funcione e interactúa entre el cliente y los gestores.
- **Equipo de Desarrollo:** Realizan las tareas del Product Backlog.

Gallinas

No son parte del proceso, pero se requiere su revisión y retroalimentación.

- **Usuarios:** Es el adquiriente del producto.

- **Stakeholders:** Las personas beneficias del proyecto y pueden participar en las revisiones del Sprint.

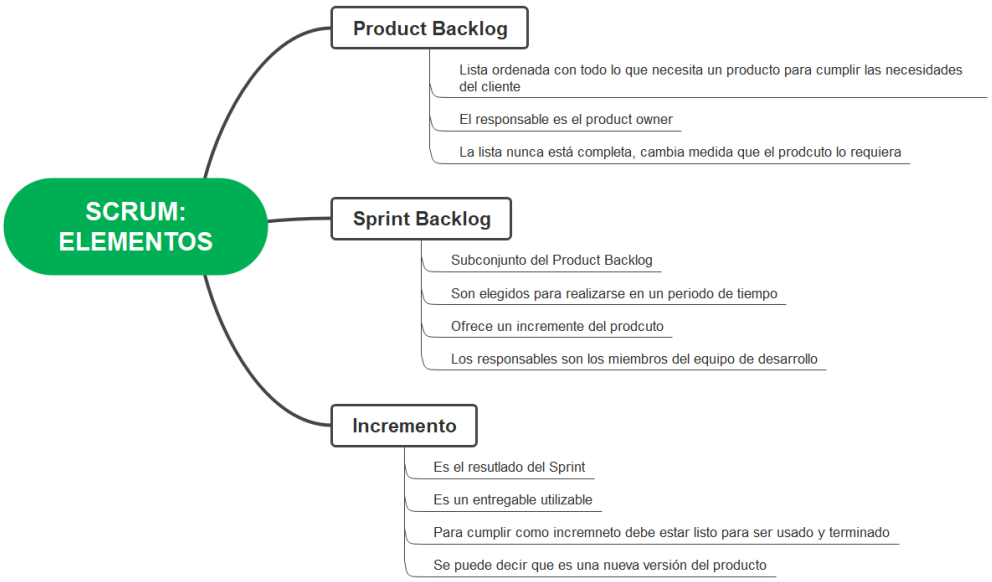
- **Managers:** Toman las decisiones finales sobre los objetivos.

Elementos

Los elementos son: Product Backlog, Sprint Backlog e Incremento.

Figura 3

Elementos de Scrum.



Product Backlog

Almacena todos los requisitos en forma priorizada, estos se irán procesando o culminando según avance el proyecto y las iteraciones.

Para iniciar el proyecto, el primer Sprint, se debe tener detallado los objetivos y la lista de requisitos y se tendrá que acordar cuando un objetivo está completo, entendiéndose por completo cuando en el entregable se verifique que los requisitos se han cumplido.

El Product Backlog no termina mientras el producto se encuentre en el mercado, es la manera en la que un proyecto evoluciona y tener un producto de valor para el cliente.

Historias de Usuario.

Son la descripción de las funciones y/o requerimientos del producto. Las historias de usuario se generan de la interacción entre el equipo de desarrollo y el cliente y evolucionan constantemente a lo largo del ciclo de vida del proyecto.

Para realizar una historia de usuario se debe tomar en cuenta los siguientes datos:

- **Número:** Identificador de la historia de usuario.
- **Usuario:** Usuarios involucrados en la historia de usuario.
- **Nombre de la historia:** Un título descriptivo para la historia de usuario.
- **Prioridad del Negocio:** Prioridad de la historia de usuario con respecto al resto.
- **Riesgo en Desarrollo:** Estimación de que aparezcan impedimentos al momento del desarrollo.
- **Puntos estimados:** Estimación relativa de la dificultad de la historia de usuario.
- **Iteración asignada:** Número de iteración a la que está asignada la historia de usuario.
- **Descripción:** Resumen de la historia de usuario.
- **Programador responsable:** El nombre del programador del equipo de desarrollo responsable de la historia de usuario.

- **Validación:** Criterio de validación para considerar la historia de usuario con la funcionalidad completa.

Tabla 1

Historia de Usuario

Historias de Usuario			
Número: 1	**Usuario:**	Cliente	
Nombre historia:	Historia 1		
Prioridad en negocio:	Alta	**Riesgo en Desarrollo:**	Baja
Puntos Estimados:	2	**Iteración Asignada:**	1
Programador Responsable:		Juan José Pérez	
Descripción:	Descripción del requerimiento del cliente		
Validación:	Validación del requerimiento del cliente		

Formato de la pila del Product Backlog

La información a colocar en la lista es a decisión del equipo de desarrollo con el Scrum Máster, y debería tener al menos la siguiente información:

- Identificador para la funcionalidad.
- Descripción de la funcionalidad.
- Sistema de priorización.
- Estimación.

Tabla 2

Prioridades de estimación

ID	PRIORIDAD	DESCRIPCIÓN	ESTIMACIÓN
1	Alta	Requerimiento 1	2
2	Baja	Requerimiento 1	1
3	Media	Requerimiento 1	4
4	Alta	Requerimiento 1	1

Sprint Backlog

Es la lista de tareas que elabora el equipo de desarrollo para la ejecución del Sprint, se asignan tareas a cada desarrollador y una estimación del tiempo para terminarlas. De esta manera el proyecto de divide en mini proyecto facilitando el encontrar donde existen inconvenientes de desarrollo e intentar resolverlos.

Se debe incluir lo siguiente:

- Lista de tareas
- Desarrollador responsable
- Estado
- Estimación

Cada desarrollador puede consultar diariamente la pila de tareas y permite tener una referencia del tiempo de cada una.

Tabla 3:

Sprint Backlog

REQUISITO	TAREA	DESAROLLADOR	ESTADO	ESTIMACIÓN(HORAS)
Requisito A	Tarea 1	Juan José Pérez	Completada	16
Requisito A	Tarea 2	Juan José Pérez	Completada	12
Requisito A	Tarea 3	Juan José Pérez	En progreso	8
Requisito A	Tarea 4	Juan José Pérez	No Iniciada	8
Requisito B	Tarea 5	Juan José Pérez	No Iniciada	16
Requisito B	Tarea 6	Juan José Pérez	No Iniciada	4
Requisito C	Tarea 7	Juan José Pérez	No Iniciada	16

Incremento

Son todos los requisitos operativos que se han completado en la iteración.

WebSockets

Es una tecnología Web en la especificación HyperText Markup Language, versión 5 (HTML5) que proporciona canales de comunicaciones full-dúplex a través de una única conexión TCP (Fette, 2011). De esta forma, el protocolo WebSocket posibilita una mayor interacción entre un navegador y un servidor Web. Sin embargo, los navegadores heredados carecen del soporte de la tecnología WebSocket emergente.

Normalmente por seguridad los administradores de redes de datos bloquean las conexiones TCP de puertos distintos al 80, y el uso de websockets aparece como una solución a esta limitación brindando una funcionalidad similar a la apertura de múltiples conexiones en distintos puertos, con la diferencia de un único puerto TCP puede multiplexar varios servicios WebSocket (Demo, Painefilu, Ferreira Szpiniak, & Zorzán, 2014).

Para implementar un cliente WebSocket se utiliza el método HTTP CONNECT estableciendo un túnel de comunicación constante entre servidor-cliente, el usuario se conecta configurando un proxi a un host y puerto remoto (WebSocket.org, 2015). El protocolo está definido con los prefijos ws:// para Websocket y wss:// para WebSocket Secure (Fette, 2011).

Cuando se requiera una conexión WSS es obligatorio la utilización de una capa TLS, con esto la data enviada mediante HTTP CONNECT se remitirá cuando el navegador se encuentre habilitado para el uso del servidor proxy explícito. Entre el cliente WSS y el servidor websocket se entabla un túnel de comunicación TCP punto a punto de bajo nivel. Existen otros casos en los que el servidor proxy es transparente y en este el navegador desconocerá su uso, por lo tanto, no en enviará la petición HTTP CONNECT. El utilizar WSS conlleva un consumo de recursos alto.

Tiempo Real

La web en tiempo real es una red web que utiliza tecnologías y prácticas que permiten a los usuarios recibir información tan pronto como es publicada por sus autores, en lugar de exigir que ellos o su software comprueben una fuente periódicamente para actualizaciones (Fette, 2011).

Un software en tiempo real es el que replica cada estímulo de la interacción con el entorno físico en un lapso de tiempo determinado (Burns, 2009). Tiene que ejecutarse en los plazos planteados y las acciones del sistema deben ser correctas.

Existen dos tipos, tiempo real duro y suave.

- Tiempo Real Duro: También conocido como crítico, los plazos de tiempo deben cumplirse y un retardo puede ocasionar fallos en lo requerido por el cliente.
- Tiempo Real Suave: Ocasionalmente los plazos pueden aplazarse sin afectar el funcionamiento.

Las características del software en tiempo real son las siguientes:

- Determinismo

Se trata de saber exactamente cómo funciona el entorno de su sistema. Asegúrese de que el sistema no deje de responder por completo. Esto es importante porque los sistemas en tiempo real deben realizar ciertas tareas antes de que se activen otros sistemas. Es importante saber esto porque casi todas las solicitudes de interrupción son generadas por eventos fuera del sistema (es decir, solicitudes de servicio). Por lo tanto, es importante especificar cuánto tiempo tarda el sistema en aceptar esta solicitud de servicio.

- Responsividad

Cuando los resultados de los cálculos determinísticos y reactivos están disponibles, se convierten en una característica del sistema y los requisitos para las aplicaciones que se ejecutan en él (por ejemplo, se completan en un momento determinado, por lo que es necesario asegurarse de que las tareas realizadas por nuestra aplicación no caen en el estado 5 (baja potencia).

Los aspectos a los que se enfoca son:

  - Tiempo que tarda en empezar a ejecutarse la interrupción.
  - Tiempo empleado en ejecutar la tarea que provocó la interrupción.
  - Efectos de las interrupciones anidadas.

- Usuarios controladores

Estos sistemas permiten que el usuario (es decir, los procesos que se ejecutan en el sistema) tenga más control sobre el sistema.

- Un proceso puede especificar su prioridad.
- Los procesos pueden especificar la gestión de memoria que desean (qué partes se almacenan en caché, qué partes se intercambian y utilizan los algoritmos, cómo se calcula el espacio de intercambio).
- Cómo especificar los privilegios que tiene en su sistema.

Aunque esto pueda sonar anárquico, no lo es, porque los sistemas de tiempo real utilizan Tipos de Proceso que ya incluyen estas características y normalmente estos TIPOS de proceso se enumeran a continuación. Un ejemplo es:

Las rutinas de mantenimiento no deben exceder el 3 % de la capacidad de la CPU a menos que en el momento en que se ejecutan el sistema se encuentre en su punto más bajo de uso. (Operativos, 2017)

- Confiabilidad

La confiabilidad del sistema en tiempo real es otra característica importante. No solo el sistema debe estar libre de errores, sino que la calidad de los servicios proporcionados no debe degradarse más allá de ciertos límites.

Un sistema debe seguir funcionando en caso de desastre o falla mecánica. La degradación del servicio en los sistemas en tiempo real a menudo tiene consecuencias desastrosas.

- Operación a prueba de fallas duras (fail hard operation)

El sistema debe fallar de tal manera que se conserve la mayor cantidad posible de datos y espacio de almacenamiento en el sistema en caso de error.

El sistema es estable. En otras palabras, si el sistema no puede completar todas las tareas dentro del límite de tiempo, el sistema completará las tareas más importantes con la prioridad más alta.

Trabajos Relacionados

Entre los trabajos relacionados, el trabajo propuesto por Ma y Sun (2013) muestra como la utilización de Websockets en un sistema disminuye el tiempo de latencia y tienen baja tasa de redundancia, esto facilita la utilización de aplicaciones en Tiempo Real.

"La comunicación por Internet proporciona un intercambio de información conveniente, hipervínculo, sin estado, pero puede ser problemático cuando se necesita el intercambio de datos en tiempo real. El protocolo WebSocket reduce la sobrecarga de comunicación de Internet y proporciona una comunicación eficiente y con estado entre los servidores Web y los clientes." (Pimentel & Nickerson, 2012)

Fernández y Sanchéz (2015) en su artículo "Aplicacions en temps real mitjançant l'ús de web sockets en PHP5" detalla las dependencias y limitaciones a la hora de solucionar aplicaciones en tiempos real con el lenguaje PHP5. Soluciona las problemáticas de dinamismo e intercomunicación de usuarios con el objetivo de mantener siempre la página actualizada a todas las conexiones al sistema.

Según Puranik, Feiock y Hill (2013) "Los resultados de nuestro estudio muestran que un servidor WebSockets consume un 50% menos de ancho de banda que un servidor AJAX; Un cliente de WebSockets consume memoria a tasa constante, no a un ritmo creciente; Y WebSockets puede enviar hasta 215,44% más muestras de datos al consumir la misma cantidad de ancho de banda de red que AJAX."

Analizando el artículo "Análisis del uso de Websockets para implementar aplicaciones web en tiempo real" (Mendoza, 2017), podemos llegar a las siguientes conclusiones:

- Las aplicaciones de software en tiempo real se caracterizan por el hecho de que las respuestas a estímulos o eventos ocurren en intervalos de tiempo específicos y, al mismo tiempo, están restringidas a las demandas del entorno.

- Lo que todas estas aplicaciones tienen en común es la necesidad de mostrar información de manera oportuna después de que hayan ocurrido eventos y entidades de escucha.

- La transmisión del protocolo HTTP es "Half dúplex". En otras palabras, es bidireccional, pero solo se puede enviar en una dirección a la vez, por lo que el cliente debe esperar la respuesta del servidor antes de realizar otra petición. AJAX es una tecnología diseñada para hacer que la web se vea más dinámica.

- Otra gran necesidad actual, especialmente en aplicaciones IoT (Internet de las cosas), es que la comunicación no necesariamente la inicia el cliente, sino el servidor como resultado de un evento generado por una señal de un dispositivo externo. , necesito recibir información y enviarla al cliente para mostrarla en el navegador.

- WebSocket es un protocolo de capa de transporte TCP. La propiedad de ser bidireccional, es decir, el servidor puede enviar mensajes en cualquier momento y el cliente también puede enviar mensajes.

- Para aplicaciones en tiempo real que requieren baja latencia entre el cliente y el servidor. B. Para aplicaciones de juegos en línea, sistemas de monitoreo de señales de dispositivos o sistemas que monitorean información que cambia constantemente, como noticias o páginas de visualización de marcadores de juegos, el protocolo WS da el contexto adecuado para respuestas exigentes brindando los mejores tiempos de respuesta para interactuar con exigencias de demanda.

La siguiente información es tomada del artículo “WebSocket Adoption and the Landscape of the Real-Time Web” (Murley, 2021):

- La web moderna se ha expandido a una amplia gama de aplicaciones que requieren actualizaciones bidireccionales entre partes: los juegos en línea, la publicidad y la edición colaborativa de documentos son algunos ejemplos.

- Los WebSockets se diseñaron para ofrecer a los desarrolladores una solución integrada y eficaz para aplicaciones en tiempo real. La API de WebSocket aborda directamente estas necesidades, proporcionando gastos generales bajos y comunicación dúplex entre los servidores web y sus clientes, evitando grandes cantidades de tráfico innecesario.

- Cuando los sitios web adoptan WebSockets, deben tener en cuenta las mejores prácticas que crearán aplicaciones más seguras y estables. WS brinda un beneficio general significativo a los usuarios en la web y abogamos por una adopción ampliada, junto con una documentación mejorada.

En el artículo “Real-time monitoring using AJAX and WebSockets” (Gupta, 2020)se realiza una comparación de estas dos tecnologías, y destaca notablemente la utilización de websockets cuando se requiere un software en tiempo real:

- AJAX significa JavaScript asíncrono y XML. XMLHttpRequest(XHR) es una API que JavaScript, JScript, VBScript y otros lenguajes de secuencias de comandos de navegadores web pueden usar para transferir y manipular datos XML hacia y desde un servidor web mediante HTTP, creando un canal de conexión independiente entre el cliente -lado y el lado del servidor de una página web.

- La API de WebSocket [9] es una tecnología avanzada que permite abrir una sesión de comunicación bidireccional interactiva entre el navegador del usuario y un servidor. Existe

una conexión persistente entre el cliente y el servidor y ambas partes pueden comenzar a enviar datos en cualquier momento.

- WebSocket es más adecuado que el polling y Comet para la comunicación en tiempo real basada en la web. Desde el punto de vista de la seguridad, tanto el protocolo WebSocket como el protocolo HTTP pueden realizar una transmisión segura. El wss y el https son sus protocolos de transmisión seguros separados. Por lo tanto, Web Socket se considera la tecnología ideal para la comunicación en tiempo real en el aspecto del tráfico, la latencia y la seguridad de la red.

- Comparación de tiempo de conexión: Al ser una solicitud, WS fue superior en tiempo a AJAX, a medida que aumentan las solicitudes AJAX maneja un tiempo mayor. Esto sucede porque webscocket abre la comunicación y la mantiene, mientras que AJAX abre y cierra la conexión.

- Comparación del ancho de banda: La propiedad del ancho de banda de red se ocupa de evaluar cuánto ancho de banda de red utilizan AJAX y WS. Seleccionamos esta propiedad debido a la economía. Esto implica que la congestión de la red puede convertirse fácilmente en un problema y provocar un retraso en la solicitud. Lo probamos en 2 Mb/s, 1 Mb/s 256 Kb/s y 512 Kb/s en todos los Socket.io superan a AJAX.

"Comunicación en tiempo real por medio de websockets para el control del prototipo Remington" (Ramirez Claros, 2020) es un artículo que analiza la visualización de la información enviada por un prototipo en forma de vehículo denominado Remington.

- Con la necesidad de tener una comunicación real, los ingenieros informáticos desarrollaron diferentes formas de lograr esto, una de las más eficientes es el método de websocket para la comunicación eficiente por medio de softwares o páginas web.

- Tras la implementación de websockets en el mundo de las redes sociales, los ingenieros han consolidado conocimientos para poder controlar dispositivos por medio de internet creando una nueva área de conocimiento llamado IOT (internet of things), en la cual el objetivo es tener comunicación en tiempo real con un dispositivo y contralarlo desde cualquier parte del mundo.

- El servidor tornado cuenta con la cualidad de adjuntar la página web y el código del websocket, el servidor encarga de analizar los datos provenientes del websocket y realiza una función específica determinada por el dato recibido.

- Una conexión de internet lenta afecta a gran modo la comunicación, haciendo que la latencia aumente y posteriormente produce retrasos en él envió de datos hacia al servidor.

Capítulo III: Especificación de Requerimientos.

En este capítulo se realizará el análisis y diseño para el desarrollo de un sistema web de comunicación para gasolineras mediante WebSockets en tiempo real, todo el contenido ha sido elaborado teniendo en cuenta las necesidades de la empresa Gasintec S.A.

Propósito

Detallar los requerimientos de software, funcionales y no funcionales de un sistema web de comunicación para gasolineras mediante WebSockets en tiempo real.

Alcance

El desarrollo de un sistema web de comunicación para gasolineras mediante WebSockets en tiempo real en Gasintec S.A. será realizada con la metodología SCRUM, y tendrá el siguiente proceso:

- Analizar y desarrollar los módulos de conexión WebSocket e ingreso de información de clientes.

  - Se mostrarán los lados de cada surtidor de despacho que tiene la gasolinera.
  - Cada lado tendrá el valor de la venta en efectivo, el volumen despachado, el precio unitario del artículo y la verificación si ya ingresó datos para la venta.
  - Conectar el sistema web con el sistema de control de los surtidores que proporcionará Gasintec S.A.
  - Al presionar en un lado se mostrará una ventana nueva con la siguiente información:
    - Código
    - Identificación
    - Nombre

- Placa
- Lista de Placas
- Forma de Pago
- Tipo de Venta

- Se buscará la información de un cliente nuevo por placa e identificación, el web service de búsqueda lo proporcionará Gasintec S.A.
- Se buscará la información de un cliente antiguo por código, placa, identificación y nombre.
- Una vez ingresada la información esta no podrá ser borrada, solo se puede actualizar.

- Verificar el funcionamiento del sistema en tiempo real y análisis de resultados.

- Verificar que el sistema cumpla con las características del tiempo real.

Definiciones, Acrónimos y Abreviaturas.

Definiciones

- **Administrador:** Usuarios con pleno acceso a toda la aplicación para realizar todas las actividades dentro del sistema.
- **Datos:** Los datos son una representación simbólica (numérica, alfabética, algorítmica, etc.), atributo o característica de una entidad. Los datos son hechos que describen sucesos y entidades.
- **Base de Datos:** Es un conjunto de datos almacenados entre los que existen relaciones lógicas y ha sido diseñada para satisfacer los requerimientos de información de una organización, almacenando en ella su descripción.

- **Cliente:** Persona que solicita el desarrollo de un producto de software.
- **Consulta:** Interrogación realizada a una base de datos, en la que se requiere una información o informaciones concretas en función de unos criterios de búsqueda definidos.
- **Desarrollador:** Profesional dedicado al desarrollo de aplicaciones de software.
- **Factibilidad:** A la opinión técnica respecto a la posibilidad de llevar a cabo una actividad determinada.
- **Funcionamiento:** Conjunto de propiedades que describen la disponibilidad y los factores que la condicionan: fiabilidad y logística de mantenimiento.
- **Hardware:** Conjunto de los componentes que integran la parte material de una computadora.
- **Interfaz:** Conexión física y funcional entre dos aparatos o sistemas independientes.
- **Interface:** Conexión e interacción entre hardware, software y el usuario. El diseño y construcción de interfaces constituye una parte principal del trabajo de los ingenieros, programadores y consultores.
- **Red:** Conjunto de computadoras interconectadas, para compartir recursos.
- **Requerimiento:** La necesidad o exigencia que el cliente espera del software.
- **Sistema Operativo:** Programa o conjunto de programas que efectúan la gestión de los procesos básicos de un sistema informático, y permite la normal ejecución del resto de las operaciones.
- **Software:** Conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora.

- **Usuario:** Persona que hace uso del módulo de control de ingreso y salida de materiales de construcción.
- **Diseño:** Actividad creativa que tiene por fin proyectar objetos, tipografías, logotipos, etc. para después fabricarlos.
- **Despachador:** Empleado de una gasolinera que realiza la venta de combustible.
- **Surtidor:** También conocido como dispensador de combustible, es una máquina de una gasolinera que se utiliza para colocar combustible en automóviles mediante mangueras.

Abreviaturas

- RF: Requisito Funcional.
- RNF: Requisito no funcional.
- NR: Nombre del requerimineto.

Referencias

- IEEE-830-1998 Software Requirements Specifications.
- Reunión con Gasintec S.A.

Perspectivas del Producto

Una estación de servicio se compone de varios aspectos para su correcto funcionamiento: eléctrica, hidráulica, ambiental, contable y de facturación. En este proyecto nos centraremos en el control de la facturación. Los surtidores de gasolina son los que emiten la información de lo vendido, tanto en volumen como en dinero. La mayoría de propietarios de gasolineras se quejan por el tiempo de venta, la demora al ingresar información del cliente o las alternativas de ingreso.

El encargado del ingreso de información y cobro es el despachador, mientras más facilidades se le conceda en el software, relativamente menos errores cometerá y tendrá un despacho más ágil.

El realizar un sistema web de comunicación para gasolineras mediante WebSockets en tiempo real es con el objetivo de tener un mejor control del proceso de venta de combustibles, proporcionando al usuario(despachador) una herramienta amigable e intuitiva que mejore el tiempo de venta.

Funciones del Producto

Cuando el sistema se encuentre en funcionamiento tendrá las siguientes funciones:

- Para el inicio de sesión, cada despachador ingresará con su usuario.
- La ventana de control de ventas mostrará todos los surtidores y lados activos e inactivos.
- En cada lado se mostrará la siguiente información:

  - Número de lado (1,2,3,4, etc.)
  - Valor en volumen con 3 decimales.
  - Valor del precio unitario con 3 decimales.
  - Valor de la venta total con 2 decimales.
  - Tipo de Venta
    - Factura
    - Crédito

- Para el ingreso de la información el despachador deberá presionar en el lado.
- Por defecto la primera búsqueda será por placa, y existirá un desplegable para escoger otra opción como código, identificación o nombre.

- La búsqueda se realizará a la base de datos propia de la gasolinera, de no existir se realizará en una base en común de clientes proporcionada por Gasintec S..A.
- Si el cliente tiene más de una placa se listará para que el despachador la seleccione.
- Se mostrará en pantalla en tiempo real el valor despachador por el surtidor de gasolina y se identificará los distintos estados por colores.

  - Inactivo – Plomo
  - Activo – Blanco
  - Despachando – Morado
  - Cerrado – Rojo

Características de los usuarios

- Despachador

  - **Descripción:** Tiene acceso a visualizar la información de los surtidores y al ingreso de datos de los clientes.
  - **Responsabilidades.** Registrar los datos del cliente.
  - **Criterio de éxito:** Se muestran los datos de la venta en el lado realizado.

Restricciones

- Para buscar en la base de datos proporcionada por Gasintec S.A. la estación de servicio debe tener internet.
- Si uno o más surtidores no se encuentran conectados al sistema no se mostrará la venta realizada.

Interfaces de Software

- Lenguajes de Programación

  - Java
  - Primefaces
  - Javascript

- Framework

  - JSF

- Herramienta de Programación

  - Apache Netbeans IDE 12.1

- Base de datos

  - PostgreSQL

- Sistema Operativo

  - Windows Server 2016/Windows 7

- Servidor

  - Wildfly

Historias de Usuario

Tabla 4

Historia de Usuario 1

Historias de Usuario			
Número: 1	**Usuario:** Despachador		
Nombre historia:	Login		
Prioridad en negocio:	Alta	**Riesgo en Desarrollo:**	Baja
Puntos Estimados:	1	**Iteración Asignada:**	1
Programador Responsable:	Juan José Pérez		
Descripción:	El sistema ofrecerá al usuario una pantalla para ingresar su usuario y contraseña.		
Validación:	El despachador ingresará el usuario y la contraseña asignada e ingresarán al sistema.		

Tabla 5

Historia de Usuario 2

Historias de Usuario			
Número: 2	**Usuario:** Despachador		
Nombre historia:	Visualización de lados de los surtidores		
Prioridad en negocio:	Alta	**Riesgo en Desarrollo:**	Media
Puntos Estimados:	2	**Iteración Asignada:**	1
Programador Responsable:	Juan José Pérez		
Descripción:	Luego del login, el despachador observará todos los lados de cada surtidor de la gasolinera con la información de ventas realizadas o realizándose.		
Validación:	Visualización en tiempo real de la información de los surtidores de gasolina.		

Tabla 6

Historia de Usuario 3

Historias de Usuario			
Número: 3	**Usuario:** Despachador		
Nombre historia:	Cambio de colores.		
Prioridad en negocio:	Alta	**Riesgo en Desarrollo:**	Baja
Puntos Estimados:	2	**Iteración Asignada:**	1
Programador Responsable:	Juan José Pérez		
Descripción:	Se mostrará cada estado de la venta en distintos colores para diferenciarlos.		
Validación:	Los colores y estados son los siguientes: ○ Inactivo – Plomo ○ Activo – Blanco ○ Despachando – Morado ○ Cerrado – Rojo		

Tabla 7

Historia de Usuario 4

Historias de Usuario			
Número: 4	**Usuario:** Despachador		
Nombre historia:	Información del lado		
Prioridad en negocio:	Alta	**Riesgo en Desarrollo:**	Media
Puntos Estimados:	1	**Iteración Asignada:**	2
Programador Responsable:	Juan José Pérez		
Descripción:	En una ventana visualizar la información almacenada en la base de datos de un lado específico.		
Validación:	Al presionar en un lado, este desplegará información como los datos del cliente, la forma de pago, el tipo de venta, la placa del vehículo.		

Tabla 8

Historia de Usuario 5

Historias de Usuario			
Número: 5	**Usuario:** Despachador		
Nombre historia:	Registro de Clientes		
Prioridad en negocio:	Alta	**Riesgo en Desarrollo:**	Media
Puntos Estimados:	2	**Iteración Asignada:**	2
Programador Responsable:	Juan José Pérez		
Descripción:	Al visualizar la información del lado, el despachador registrará o actualizará la información del cliente		
Validación:	Ingreso y búsqueda de la información del cliente y almacenamiento en la base de datos.		

Tabla 9

Historia de Usuario 6

Historias de Usuario			
Número: 6	**Usuario:** Despachador		
Nombre historia:	Búsqueda de clientes nuevos		
Prioridad en negocio:	Media	**Riesgo en Desarrollo:**	Media
Puntos Estimados:	2	**Iteración Asignada:**	2
Programador Responsable:	Juan José Pérez		
Descripción:	Si un cliente no está registrado en la base de datos de la gasolinera se procederá a consultar en una base de datos en común mediante un servicio web.		
Validación:	Al enviar la solicitud al servicio web este devolverá la información del cliente en json y se mostrará en una ventana emergente.		

Requerimiento Funcionales

En la siguiente tabla se puede observar los requerimientos funcionales.

Tabla 10

Requerimientos Funcionales

RF	**NR**	**Características**
RF-01	Login	El sistema ofrecerá al usuario una pantalla para ingresar su usuario y contraseña.
RF-02	Visualización de lados de los surtidores	Luego del login, el despachador observará todos los lados de cada surtidor de la gasolinera con la información de ventas realizadas o realizándose.
RF-03	Cambio de colores.	Se mostrará cada estado de la venta en distintos colores para diferenciarlos.
RF-04	Información del lado	En una ventana visualizar la información almacenada en la base de datos de un lado específico.
RF-05	Registro de Clientes	Al visualizar la información del lado, el despachador registrará o actualizará la información del cliente
RF-06	Búsqueda de clientes nuevos	Si un cliente no está registrado en la base de datos de la gasolinera se procederá a consultar en una base de datos en común mediante un servicio web.

Requerimientos No Funcionales

En la siguiente tabla se puede observar los requerimientos no funcionales.

Tabla 11

Requerimientos No Funcionales

RNF	**NR**	**Características**
RNF-01	Disponibilidad	El enlace por websocket desde la aplicación proporcionada por Gasintec debe estar siempre disponible para obtener los datos.
RNF-02	Operatividad	El sistema debe ser intuitivo y fácil de usar para los usuarios que en su mayoría no poseen conocimientos tecnológicos.
RNF-03	Fiabilidad	La información a mostrar en pantalla para el despachador debe ser la misma que se muestra en la pantalla del dispensador de combustible.

Capítulo IV: Desarrollo e Integración.

En este capítulo se desarrollará el sistema web de comunicación para gasolineras mediante WebSockets en tiempo real en Gasintec S.A. con la utilización de la metodología SCRUM, esta fue escogida con la finalidad de tener una constante comunicación con el cliente y la verificación de avances parciales de forma regular.

Levantamiento de Requerimientos

Como primer paso definimos los roles del equipo de SCRUM y las personas asignadas a cada una.

Tabla 12

Roles de equipo

Roles	**Descripción**	**Persona**
Product Owner	Gerente General Gasintec S.A.	Ing. David Herdoíza
Scrum Master	Director del proyecto	Juan José Pérez
Development Team	Desarrollador	Juan José Pérez

Para el levantamiento de requerimientos se realizaron una sucesión de entrevistas al Product Owner, siendo estas necesarias para entender el campo de acción de la empresa y lo que desean obtener del proyecto.

Estas entrevistas permitieron entender los procesos que se ejecutan en las gasolineras, así como sus actores y variables a tomar en cuenta al momento de realizar el desarrollo. Además, facilitó el desarrollo de los procesos a realizar en la aplicación siendo estos los siguientes:

- **Login:** Cada despachador ingresará con su usuario y contraseña que serán previamente proporcionadas por el/los administrador(es) de la gasolinera. Al ingresar se habilitarán las siguientes opciones:

  - **Impresión Islas:** Pantalla en la que se visualiza toda la información para la venta de los despachadores.

  - **Cerrar Turno:** Botón que se habilitará al momento de cerrar el turno.

  - **Fecha y hora de turno:** Se mostrará la fecha del turno, su horario y un reloj con la hora actualizada.

- **Visualización de lados de los surtidores:** Se mostrarán todos los lados de la gasolinera siempre y cuando la bandera Mostrar Todo se encuentre activa, caso contrario se mostrarán solo los asignados al despachador.

  - **Lado:** Será identificado con un número, cada surtidor de gasolina puede tener hasta 2 lados, y serán identificados con un número que no podrá repetirse. En la parte inferior se mostrará el volumen despachador, precio unitario del producto, y venta total actualizada.

- **Cambio de colores:** Cada color indicará el estado de la venta, y este cambiará al instante que se ejecute la acción en el surtidor. Los estados y colores serán los siguientes.

  - **Inactivo:** Será de color plomo, indica que el lado no se encuentra en funcionamiento, por lo tanto, no mostrará tampoco información de ventas.

  - **Activo:** Será de color blanco, indica que el lado está listo para que el despachador venda combustible.

  - **Despachando;** Será de color morado, indica que el lado está vendiendo, adicional del color se mostrará la venta en tiempo real.

  - **Cerrado:** Será de color rojo, indica que la venta terminó, la manguera cerró su flujo de combustible y espera que el despachador ingrese la información del cliente para pasar a estado activo.

- **Información del lado:** Para pasar de estado cerrado a activo, es obligatorio el ingreso de la información del cliente, por lo tanto, al presionar sobre el lado, se desplegará la información a ingresar por el despachador.

  - **Datos del cliente:** La información relevante y solicitada por el Servicio de Rentas Internas para su autorización como identificación y nombre. y los datos opcionales como dirección, teléfono y correo electrónico.

  - **Tipo de Venta:** Existen 4 tipos de venta para un cliente. A un cliente normal se le emite una factura, si tiene convenio para realizar consumos indefinidos para posteriormente realizarle una factura de todos estos es un cliente a Crédito, si tiene un convenio para realizar consumos hasta llegar a un valor previamente acordado, es un cliente Prepago, y si es quien realiza medición de que la cantidad despachada sea la correcta es un cliente Calibración.

  - **Forma de Pago:** Si la venta es con factura, es obligatorio el seleccionar la forma de pago, siendo estas las siguientes.

Tabla 13

Formas de Pago

Forma de Pago SRI	**Forma de Pago Sistema**
Sin utilización del sistema financiero	Efectivo
Con utilización del sistema financiero	Transferencia
	Tarjeta de Crédito
	Tarjeta de Débito
	Cheque
	Factura por cobrar

- **Búsqueda de Clientes:** Para buscar la información del cliente el despachador lo puede hacer de cuatro maneras distintas.
    - Nombre
    - Identificación
    - Placa
    - Código
- **Búsqueda de clientes nuevos y registro de placas:** Para buscar un cliente, el primer paso es revisar si existe un registro en la base de datos local, si la búsqueda es por placa o identificación y no existe ningún registro, se procederá a realizar una consulta al servicio web de Gasintec. Antes de consultar/registrar se debe tener en cuenta lo siguiente.
    - Si es cédula debe estar validada según los parámetros del servicio de rentas internas.
    - Si es R.U.C debe tener trece caracteres siendo los últimos tres 001

- Si es placa debe mantener la estructura requerida por los entes de control del Ecuador.

Dentro de las entrevistas se definió el usuario que hará uso de la aplicación, este será conocido como despachador, es la persona que realiza el proceso de venta de combustible e ingresará la información del cliente.

Product Backlog

A continuación, se detalla el producto backlog tomando en cuenta los datos recopilados de los requerimientos del cliente

Historias de Usuario

Tabla 14

Product Backlog - Historia de Usuario 1

Historias de Usuario			
Número: 1	**Usuario:** Despachador		
Nombre historia:	Login		
Prioridad en negocio:	Alta	**Riesgo en Desarrollo:**	Baja
Puntos Estimados:	2	**Iteración Asignada:**	1
Programador Responsable:	Juan José Pérez		
Descripción:	El sistema ofrecerá al usuario una pantalla para ingresar su usuario y contraseña.		
Validación:	El despachador ingresará el usuario y la contraseña asignada e ingresarán al sistema y se mostrarán sus opciones.		

Tabla 15

Product Backlog - Historia de Usuario 2

Historias de Usuario			
Número: 2	**Usuario:** Despachador		
Nombre historia:	Visualización de lados de los surtidores		
Prioridad en negocio:	Alta	**Riesgo en Desarrollo:**	Media
Puntos Estimados:	7	**Iteración Asignada:**	1
Programador Responsable:	Juan José Pérez		
Descripción:	Luego del login, el despachador observará todos los lados de cada surtidor de la gasolinera con la información de ventas realizadas o realizándose.		
Validación:	Visualización en tiempo real de la información de los surtidores de gasolina. La información a visualizar es: - Volumen despachado - Precio del combustible - Valor total de venta		

Tabla 16

Product Backlog - Historia de Usuario 3

Historias de Usuario			
Número: 3	**Usuario:** Despachador		
Nombre historia:	Cambio de colores.		
Prioridad en negocio:	Alta	**Riesgo en Desarrollo:**	Baja
Puntos Estimados:	4	**Iteración Asignada:**	1
Programador Responsable:	Juan José Pérez		
Descripción:	Se mostrará cada estado de la venta en distintos colores para diferenciarlos.		
Validación:	Los colores y estados son los siguientes: ○ Inactivo – Plomo ○ Activo – Celeste ○ Despachando – Morado ○ Cerrado – Rojo		

Tabla 17

Product Backlog - Historia de Usuario 4

Historias de Usuario			
Número: 4	**Usuario:** Despachador		
Nombre historia:	Información del lado		
Prioridad en negocio:	Alta	**Riesgo en Desarrollo:**	Media
Puntos Estimados:	2	**Iteración Asignada:**	2
Programador Responsable:	Juan José Pérez		
Descripción:	En una ventana visualizar la información almacenada en la base de datos de un lado específico.		
Validación:	Al presionar en un lado, este desplegará la siguiente información: - Datos del cliente - Forma de pago - Tipo de venta - Placa del vehículo.		

Tabla 18

Product Backlog - Historia de Usuario 5

Historias de Usuario			
Número: 5	**Usuario:** Despachador		
Nombre historia:	Búsqueda de Clientes		
Prioridad en negocio:	Alta	**Riesgo en Desarrollo:**	Media
Puntos Estimados:	2	**Iteración Asignada:**	2
Programador Responsable:	Juan José Pérez		
Descripción:	Al visualizar la información del lado, el despachador registrará o actualizará la información del cliente		
Validación:	Ingreso y búsqueda de la información del cliente y almacenamiento en la base de datos.		

Tabla 19

Product Backlog - Historia de Usuario 6

Historias de Usuario			
Número: 6	**Usuario:** Despachador		
Nombre historia:	Búsqueda de clientes nuevos y registro de placas		
Prioridad en negocio:	Media	**Riesgo en Desarrollo:**	Media
Puntos Estimados:	3	**Iteración Asignada:**	2
Programador Responsable:	Juan José Pérez		
Descripción:	Si un cliente no está registrado en la base de datos de la gasolinera se procederá a consultar en una base de datos en común mediante un servicio web.		
Validación:	Al enviar la solicitud al servicio web este devolverá la información del cliente en json y se mostrará en una ventana emergente para registrarlo.		

La información de cada historia de usuario es optimizada para facilitar las iteraciones y puede cambiar según avanza el proyecto, a continuación, se detalla las valoraciones.

- Prioridad en negocio: Alta-Media-Baja
- **Puntos estimados:** Tiempo estimado en días.

Sprint Planning

En la fase de planificación los miembros del equipo de desarrollo eligen una o varias historias de usuario para ejecutarlas en cada iteración o sprint, antes de empezar con las historias de usuario el sprint 0 será el de arquitectura del sistema.

Tabla 20

Sprint Planning

Descripción	Prioridad	Complejidad	Sprint	Duración(días)
Arquitectura del sistema	5	4	0	10
Login	5	2	1	5
Visualización de lados de los surtidores	5	5	1	30
Cambio de estados	5	5	1	10
Información del lado	4	4	2	5
Búsqueda de Clientes	4	3	2	5
Búsqueda de clientes nuevos y registro de placas	4	3	2	5
				70

La valoración dada a prioridad y complejidad son de mayor a menor, siendo 1 la prioridad menos urgente y 5 la más urgente, y también 1 menor complejidad y 5 mayor complejidad.

Sprint Backlog 0

Una vez identificados las acciones a realizar en cada iteración, procedemos a realizar la arquitectura en la cual se va a realizar el desarrollo. El sprint tuvo una duración de 10 días, laborando 4 horas por día aproximadamente.

Tras varias reuniones con el Product Owner se diagramó el proceso macro y micro deseado del proyecto y se definió el diseño de la vista principal, en la siguiente tabla se muestran las distintas tareas a realizarse y el tiempo estimado para culminar cada tarea.

Tabla 21

Sprint Backlog 0

ID	Responsable	Descripción	Tareas	Esfuerzo empleado en días
1	Juan José Pérez	Necesito entender el proceso de venta	Realizar el diagrama del funcionamiento	2
			Revisión del diagrama	1
			Realizar el diagrama de acciones	2
			Revisión del diagrama	1
2	Juan José Pérez	Necesito preparar el entorno de desarrollo	Diseño y revisión de la vista principal de la aplicación	2
			Configurar el framework de desarrollo	2
			Total:	10

Sprint Review 0

Para este artefacto se requiere el verificar los resultados del Sprint Backlog 0, a continuación se detalla su avance.

Tabla 22

Sprint Review 0

ID	Tarea	Iteración	Avance
1	Diagrama de Funcionamiento	0	100%
	Diagrama de Acciones	0	100%
2	Vista principal de la aplicación	0	100%
	Configuración del framework de desarrollo	0	100%

Entregables Sprint 0

La construcción de la arquitectura de la aplicación tiene los siguientes entregables:

- Diagrama Macro
- Diagrama Micro
- Vista principal.
- Framework de desarrollo.

Diagrama Macro

El diagrama macro contiene la idea del negocio, el funcionamiento del aplicativo a desarrollar en una estación de servicio.

Figura 4

Diagrama de funcionamiento

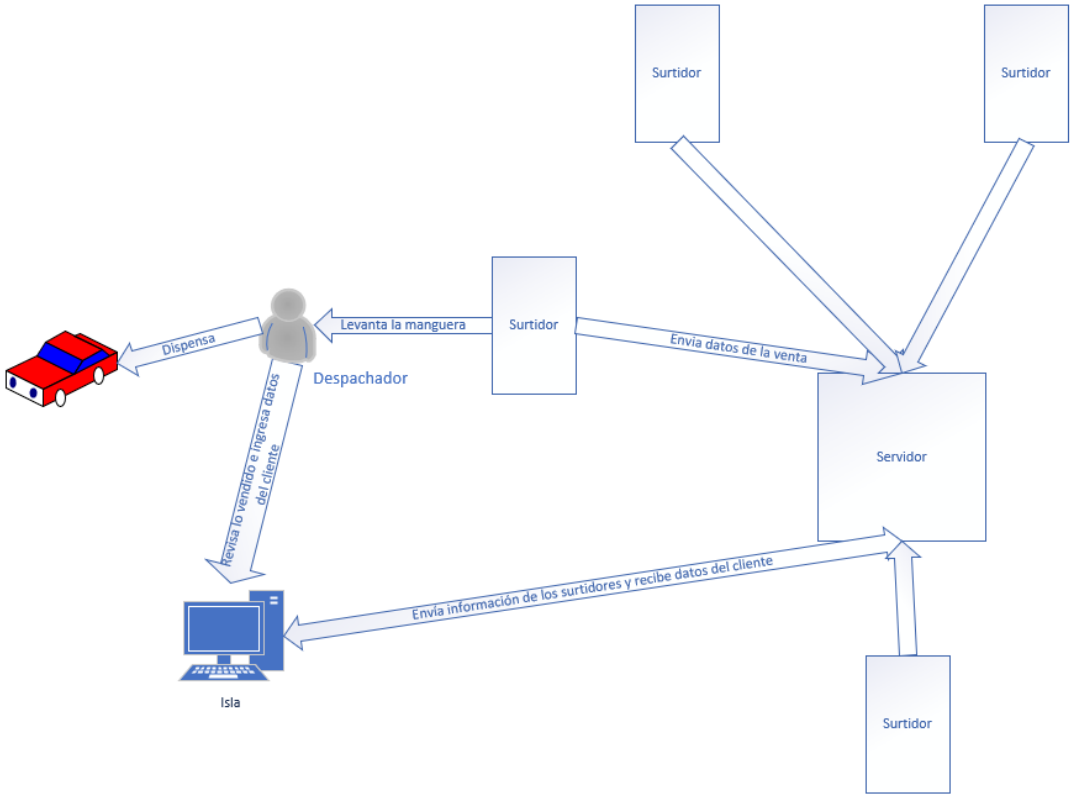
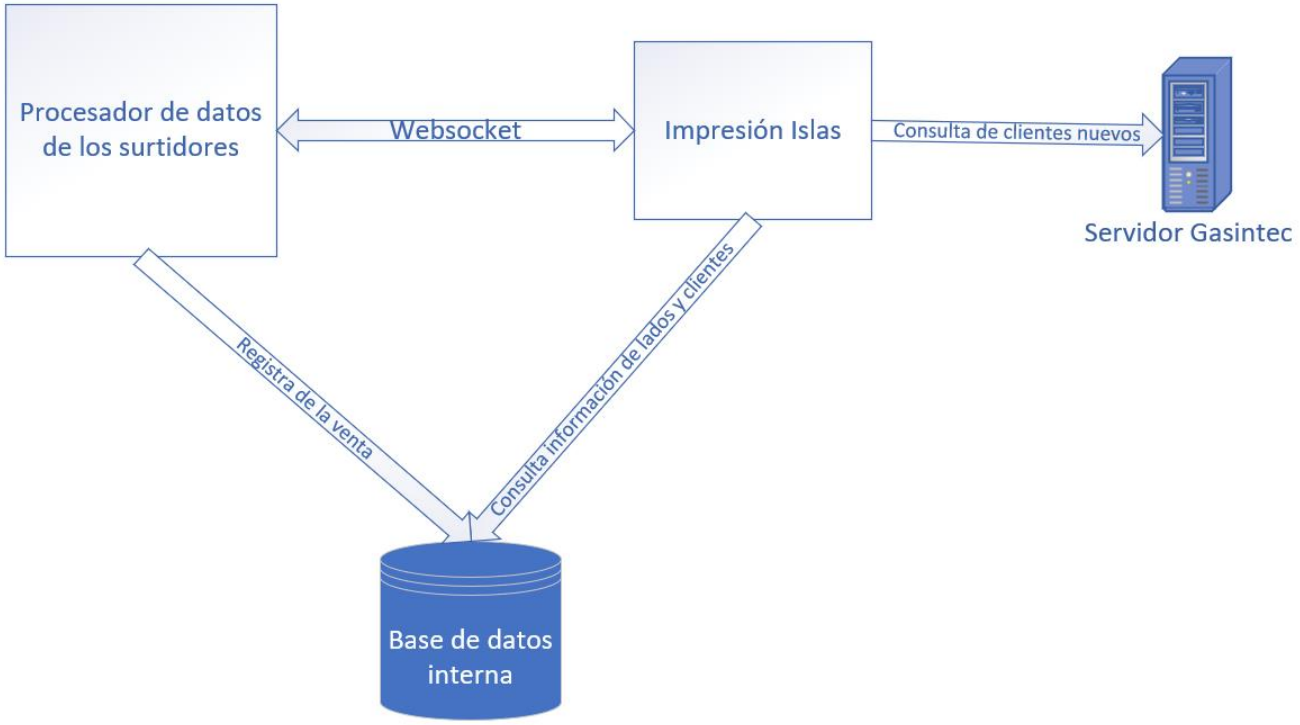


Diagrama Micro

El diagrama micro contiene todas las acciones a desarrollarse con el servidor y la base de datos.

Figura 5

Diagrama de acciones

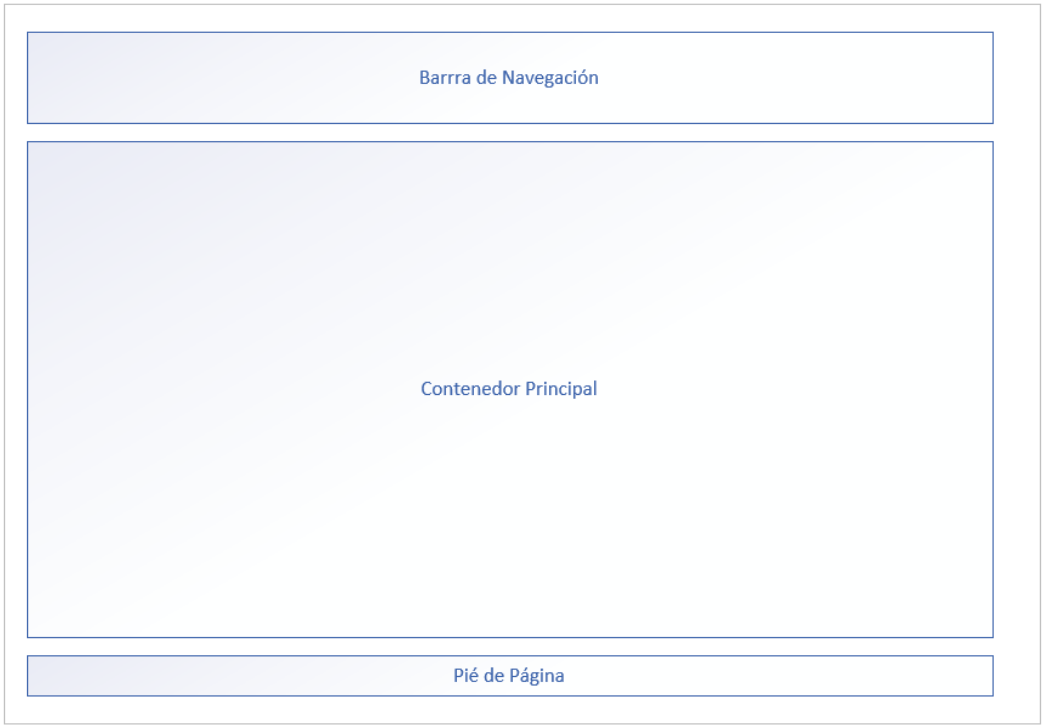


Vista Principal de la Aplicación

El diseño describe una barra de navegación en la parte superior, la cual contiene la información del usuario en sesión, el logo de la empresa, el botón de cierre de sesión y el menú que le permitirá al despachador navegar a través del aplicativo, en el contenedor principal se visualiza la información con las opciones a realizar y el pie de página presentará la información del turno y el horario del usuario.

Figura 6

Vista Principal de la aplicación



Framework de Desarrollo

Las herramientas definidas para el desarrollo del proyecto son las siguientes:

- **IDE:** Apache Netbeans IDE 12.1
- Backend: Java 8
- **Frontend:** Primefaces 10
- **Base de datos:** PostgreSQL 13
- Servidor web: Wildfly 13
- Sistema Operativo: Windows 10

Sprint Backlog 1

Una vez realizado el Sprint 0 procedemos con la ejecución de la siguiente iteración, en la iteración 1 tenemos las historias de usuario 1,2 y 3.

- Login
- Visualización de lados de los surtidores
- Cambio de colores

Tabla 23

Sprint Backlog 1

ID	Responsable	Descripción de la historia de usuario	Tareas	Esfuerzo empleado en días
3	Juan José Pérez	Necesito ingresar a la aplicación con un usuario y contraseña	Crear una vista para el ingreso	2
			Validar campos de ingreso	1
			Crear un controlador para datos de ingreso	2
4	Juan José Pérez	Necesito visualizar los lados de la estación de servicio	Crear una vista para listar los lados de los surtidores	3
			Crear un controlador para consultar los lados	2
			Conectar el websocket con la aplicación de Gasintec	10
			Mostrar la información del websocket en la vista	10
			Verificar que la información sea en tiempo real	5
5	Juan José Pérez	Necesito visualizar la información cada lado con colores distintivos	Definir los estados y colores	1
			Crear controlador de colores por estado	8
			Verificar la información en la vista.	1

Sprint Review 1

Para este artefacto se requiere el verificar los resultados del Sprint Backlog 1, a continuación se detalla su avance.

Tabla 24

Sprint Review 1

ID	Tarea	Iteración	Avance
3	Crear una vista para el ingreso	1	100%
	Validar campos de ingreso	1	100%
	Crear un controlador para datos de ingreso	1	100%
4	Crear una vista para listar los lados de los surtidores	1	100%
	Crear un controlador para consultar los lados	1	100%
	Conectar el websocket con la aplicación de Gasintec	1	100%
	Mostrar la información del websocket en la vista	1	100%
	Verificar que la información sea en tiempo real	1	100%
5	Definir los estados y colores	1	100%
	Crear controlador de colores por estado	1	100%
	Verificar la información en la vista.	1	100%

Entregables Sprint 1

La iteración 1 tiene los siguientes entregables:

- Login
- Visualización de lados de los surtidores
- Información en tiempo real de los lados de los surtidores

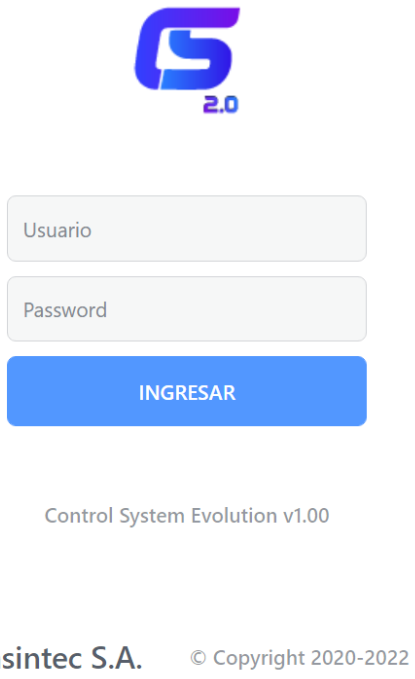
Login

Para acceder a la aplicación el despachador debe ingresar el usuario y la contraseña asignadas, caso contrario se mostrarán mensajes de advertencia y no podrá vender.

El despachador digita los datos solicitados para el ingreso, al tipear en el password solo se mostrará asteriscos (*) para ocultar la contraseña. La contraseña en la base de datos está encriptada con un algoritmo de hash seguro de 256 bits o SHA-256.

Figura 7

Vista de Login

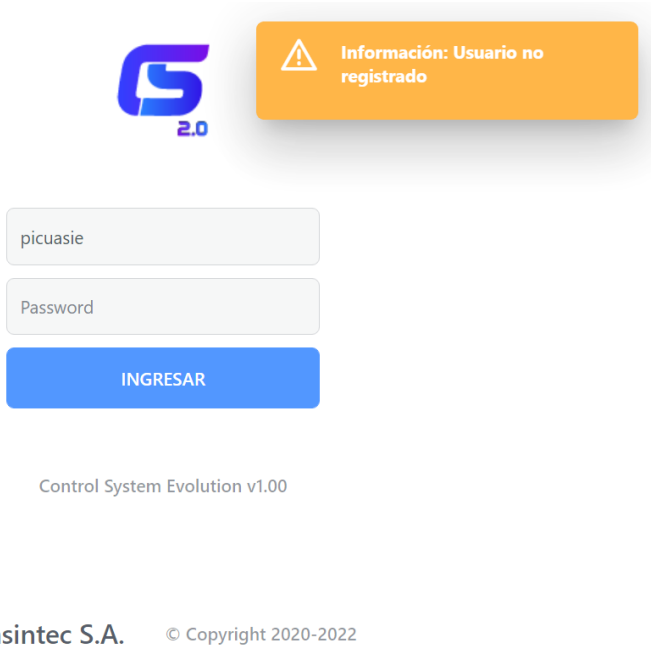


Si ingresa mal la clave o la contraseña se muestran los siguientes mensajes.

- Usuario no registrado

Figura 8

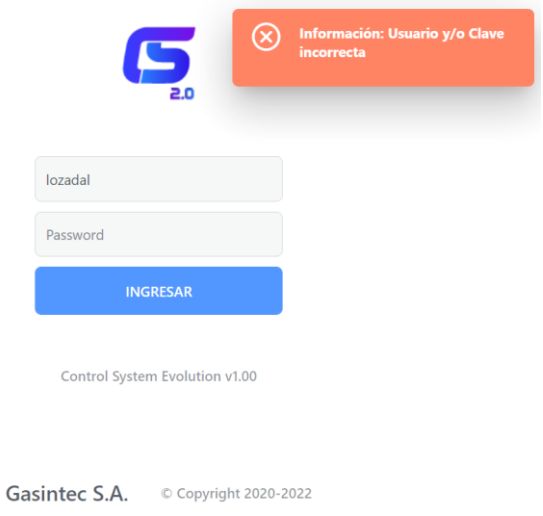
Usuario no registrado



- Usuario y/ clave incorrecta

Figura 9

Usuario y/o clave incorrecta



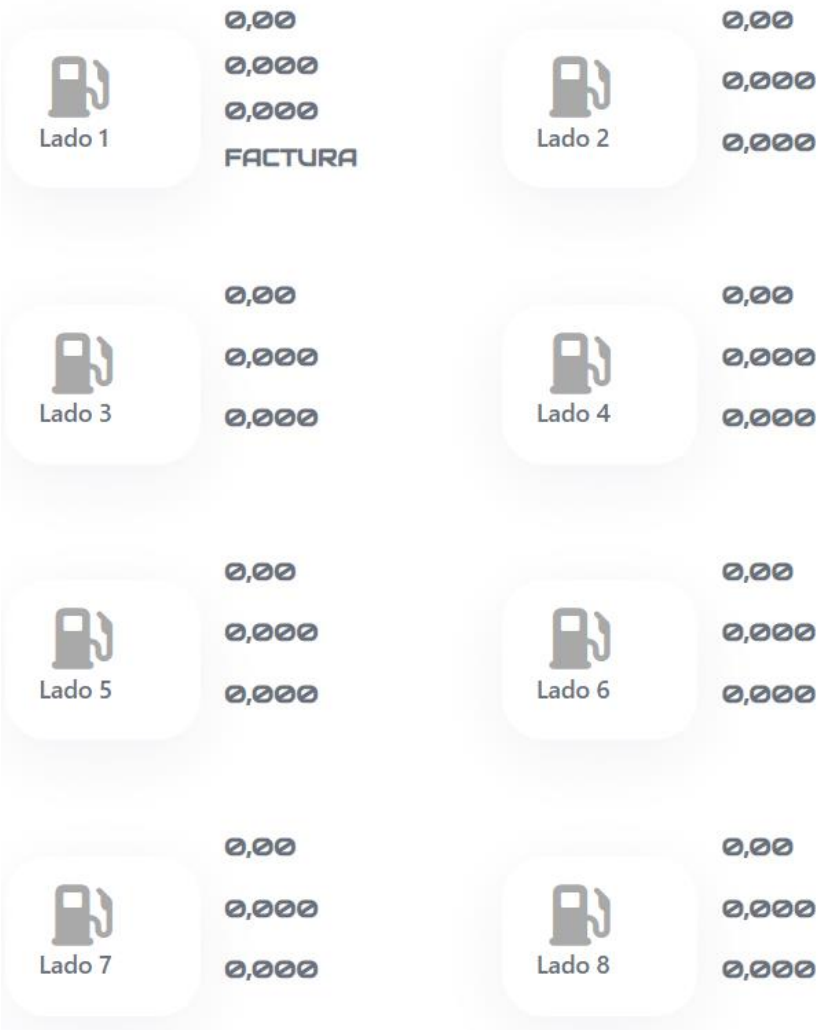
El despachador no tiene la opción de recuperar la contraseña, si se olvida el administrador del sistema le actualiza el registro.

Visualización de lados de los surtidores

El despachador ingresa al sistema y visualiza todos los lados de los surtidores de la gasolinera, un surtidor puede tener 1 o 2 lados.

Figura 10

Vista de los lados de los surtidores

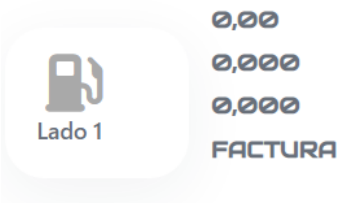


En cada lado se mostrará la siguiente información.

- Número de Lado.
- Ícono de surtidor que cambia de color según el estado de la venta.
- Total de la venta con dos decimales.
- Volumen despachado con tres decimales.
- Precio del producto despachado con tres decimales.
- Información del tipo de venta realizada en ese lado, siendo estas las siguientes:
    - Factura
    - Crédito
    - Prepago
    - Calibración

Figura 11

Información del lado



Información en tiempo real de los lados de los surtidores

Para el cambio de colores de los lados o diferenciación de estados, requerimos que la información sea proporcionada en tiempo real con el fin que el despachador despache y cobre lo justo.

Gasintec S.A. tiene un controlador que administra la información en tiempo real, para obtener esta en nuestro aplicativo web, se conectarán mediante websockets por el puerto 1000.

Se han identificado cuatro estados en los que puede estar un lado, estos son:

- **Inactivo:** El lado está registrado en el sistema, pero se encuentra apagado o está desconectado del sistema, no se puede vender de este lado. Se identifica con el color plomo.
- **Activo:** El lado está disponible para despachar combustible, los valores de las ventas están estáticas. Se identifica con el color celeste
- **Despachando:** Los valores de las ventas están en movimiento marcando lo despachado en el surtidor físico. Se identifica con el color morado.
- **Venta Cerrada:** La venta finalizó a la espera de ingresar datos del cliente. Se identifica de color rojo.

Para llegar al objetivo deseado se necesitó de varios intentos, y la utilización de varias herramientas, las cuales serán detalladas a continuación.

Poll con consulta a la base de datos

Poll es un componente AJAX de Primefaces, desde su versión 8, que realiza llamadas remotas periódicas a una función en el Java Bean.

Tomado de (Primefaces, 2022), “Primefaces es un conjunto de componentes JSF de código abierto con varias extensiones que tiene un amplio conjunto de componentes (HtmlEditor, Dialog, AutoComplete, Charts y muchos más) y con AJAX integrado basado en las API JSF AJAX estándar”.

Java Bean es el controlador de la página en el modelo desarrollado, es una clase donde se encuentra las funciones del negocio con los requisitos concretos, por cada página web existe un Java Bean y cumple con las siguientes características:

- Propiedades privadas con getters y setters
- Un constructor público sin parámetros.

Para la ejecución de lo deseado, que es mostrar la información en tiempo real, desde la página se realiza el llamado al proceso, se determina el tiempo de ejecución y los componentes a actualizar.

Figura 12

Llamado a la función transaccionar mediante el componente Poll

```
<p:poll id="plTransaccion" interval="0.5" listener="#{impresionIslasBean.transaccionar()}" update="@form"/>
```

El tiempo de ejecución está en segundos, realiza un llamado a la función “transaccionar()” que recupera la información de la base de datos y actualiza en toda la página, cada uno de los surtidores. Para esto, el controlador de Gasintec S.A. almacena toda la información de ventas en la misma base de datos.

Al ejecutar cada 0.5 segundos consultas a la base de datos existe un nivel alto de transaccionalidad, y al mostrar en la vista existía latencia ampliando este rango de 0.5 a 2.5 segundos. Otro inconveniente que presentó, por ser un intervalo corto, impedía que otras funciones del mismo Bean se ejecuten, dando por descartado esta opción. En la siguiente tabla detallo las características de tiempo real incumplidas en este proceso.

Tabla 25

Características del software en tiempo real aplicado a Poll con consulta a la base de datos

Característica	Resultado
Determinismo	- El sistema no garantizar la ejecución de otras tareas al mismo tiempo.
Responsividad	- El tiempo que tarda en ejecutar la petición afecta a la ejecución de otras. - Mientras más tareas se desean ejecutar mas lenta es la ejecución de la tarea principal.
Usuarios controladores	- El proceso de visualización de datos en tiempo real se toma el control de la vista y no prioriza actividades, no le permite al usuario realizar otros procesos.
Confiabilidad	- La calidad del servicio se degrada con el paso del tiempo.
Operación a prueba de fallas duras	- El sistema no es estable. - Si existe un fallo el sistema si preserva los datos.

El rango de pruebas fue de 0.5 hasta 2 segundos teniendo los mismos resultados. De todas las características que requiere cumplir una aplicación para determinar si se ejecuta en tiempo real, no cumple con ninguna, por lo tanto, se establece que no es factible su uso.

Poll con websocket

Para la ejecución del websocket utilizamos la librería Java WebSocket(org.java-websocket) en su versión 1.3.9. Para poder ejecutar este proceso, el controlador de Gasintec S.A. abre la conexión en un puerto específico, el sistema se conectará para recibir toda la información de los surtidores, la información recibida tiene la siguiente estructura.

Lado-producto-total-volumen-precio-estado

La información recibida es en paquetes, cada paquete contiene la información de todos los lados activos, es decir, si la gasolinera tiene diez lados y todos están funcionando correctamente recibiré diez registros en el mismo paquete, si existe un surtidor apagado o fuera de servicio, no estará en el paquete y por lo tanto el sistema no mostrará su información.

Para el proceso de visualización en tiempo real, procedimos a realizar el llamado a la función "consultarWebSocket()" en el intervalo de 0.5 segundos y se actualizará el panel lados.

Figura 13

Llamado a la función consultarWebsocket() mediante el componente Poll

```
<p:poll id="plTransaccion" interval="0.5" listener="#{impresionIslasBean.consultarWebsocket()}" update=":form:lados"/>
```

La visualización en pantalla de los lados fue más rápida comparado con el proceso anterior, pero persistía el inconveniente que permitía la ejecución de otras funciones o tareas. Si bien el observar la venta es prioritario, el registrar el cliente lo es más.

Dado que el requerimiento del cliente del sistema es ingresar la información en la misma vista, esta opción también fue descartada.

Tabla 26

Características del software en tiempo real aplicado a Poll con websocket

Característica	Resultado
Determinismo	- El sistema no garantiza la ejecución de otras tareas al mismo tiempo.
Responsividad	- La ejecución de la petición no permite la ejecución de otras. - No se puede ejecutar más de una tarea.
Usuarios controladores	- El proceso prioriza solo su ejecución
Confiabilidad	- A pesar de no ejecutarse otras tareas, luego del tiempo deseado si se visualiza el valor de las ventas.
Operación a prueba de fallas duras	- El sistema no es estable. - Si existe un fallo el sistema si preserva los datos.

El rango de pruebas fue de 0.5 segundos a 2 segundos obteniendo resultados similares, siendo confiabilidad la única característica que cumple, por lo tanto, se descarta esta opción para el proyecto.

JavaScript con websocket

JavaScript es un lenguaje de programación interpretado, es decir que no es necesario compilar los programas para ejecutarlos, que se utiliza principalmente para el desarrollo de páginas web dinámicas, no guarda ninguna relación con Java (Pérez, 2019).

Se utilizó la misma librería anterior Java WebSocket, y se enlaza con JavaScript mediante la añadidura de las etiquetas “AplicationScoped” y “ServerEndPoint” en la clase de ejecución del websocket.

Figura 14

Utilización de librerías para abrir una conexión websocket

```
@ApplicationScoped
@ServerEndpoint("/transaccion")
public class Transaccion {
```

Para abrir el websocket se necesita cuatro parámetros:

- Protocol: El protocolo es ws.
- Hostname: Es la IP dónde está localizado el controlador, por ejemplo 192.168.0.2.
- Port: El puerto en el cual se está ejecutando la página, por ejemplo 80.
- Endpoint: El enlace con la clase ejecutándose con la librería Java WebSocket, por ejemplo "/transaccion".

Figura 15

Ejemplo de creación del objeto websocket

```
webSocketURL = "ws://192.168.0.2:80/transaccion";
```

Una vez conectados, el sistema abre un puerto por el cual ingresará información sin necesidad de realizar una petición, a diferencia de las herramientas utilizadas anteriormente. La información es enviada como JSON, que contiene la data de cada lado. Por ejemplo

```
[
  {1-Extra-10-4.167-2.400-V},
  {2-Diesel Premium-0-0-1.700-D},
  {3-Super-30-5.769-5.20-L},

]
```

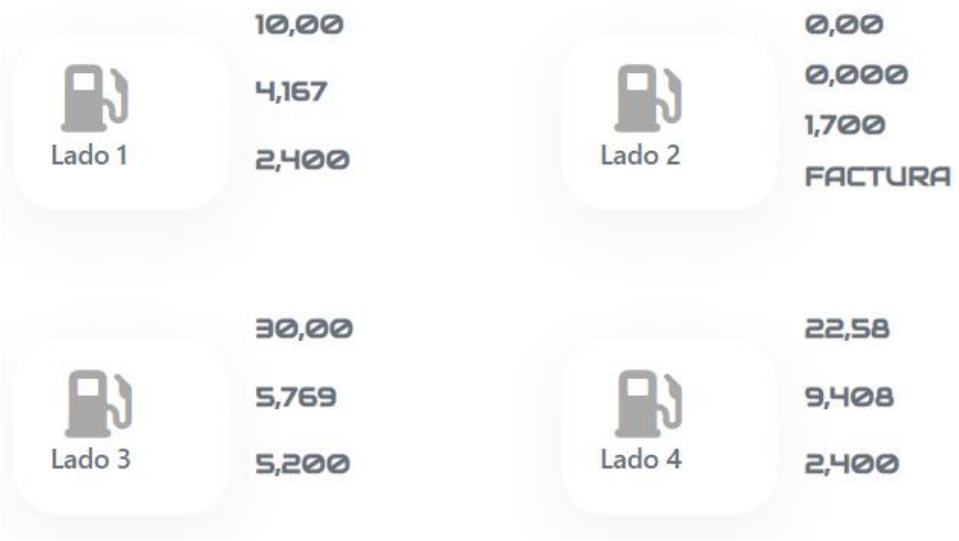
En donde,

- **V:** Venta cerrada,
- **D:** Despachando, y
- **L:** Libre o disponible

Procesamos esta información y actualizamos la vista según lo enviado.

Figura 16

Vista de los lados con información



La visualización de la información en pantalla es rápida, el procesamiento lo realiza el navegador, esto libera tareas para la ejecución de los otros procesos. Con el transcurso del tiempo, la memoria RAM

utilizada por el navegador se incrementaba exponencialmente, provocando latencia en la visualización de la información y ralentizando las otras tareas.

Tabla 27

Características del software en tiempo real aplicado a JavaScript con websocket

Característica	Resultado
Determinismo	- El sistema garantiza la ejecución de otras tareas al mismo tiempo. - El tiempo en ejecutar el procesamiento de la información es mínimo en un principio ralentizando con el trascurrir de los días.
Responsividad	- La ejecución de la petición se procesa en backgroud, por lo que no interfiere con los otros procesos, hasta que el aumento de la memoria RAM ralentiza el sistema. - El usuario al solicitar la ejecución de otras tareas no tiene efecto secundario en el procesamiento de los datos de los surtidores hasta que aumenta la memoria RAM del navegador.
Usuarios controladores	- El procesamiento de los datos de ventas se ejecuta en segundo plano, permitiéndole al usuario priorizar la tarea deseada. - El proceso no tiene control sobre la memoria RAM a utilizar y con el transcurso del tiempo almacena abundante memoria caché.
Confiabilidad	- Con el transcurso del tiempo la calidad del sistema se degrada.
Operación a prueba de fallas duras	- Si el registro de un lado está incompleto no afecta en la visualización de los otros lados, y cuando recepta el siguiente registro lo actualiza. - El sistema no es estable.

Existen varios inconvenientes para definir como que el resultado fue el deseado, siendo el mas importante la confiabilidad sobre el sistema. No existe una manera de definir la memoria RAM a utilizar por el navegador puesto que es un agente externo, se realizó pruebas en 4 navegadores distintos obteniendo los mismos resultados.

- Google Chrome
- Brave
- Opera
- Mozilla Firefox

JavaScript con websocket y API Fetch

Para obtener una solución a inconveniente de memoria, al iniciar la conexión websocket, las variables deben ser constantes, permitiendo así el uso de menos memoria en el navegador y no opacando la ejecución de otros procesos, así como tener un mejor manejo del uso de los recursos. Para esto se utilizó la función API Fetch de JavaScript.

“La API Fetch proporciona una interfaz JavaScript para acceder y manipular partes del canal HTTP, tales como peticiones y respuestas. También provee un método global fetch() (en-US) que proporciona una forma fácil y lógica de obtener recursos de forma asíncrona por la red” (MDN, 2022).

El objeto que devuelve la ejecución de esta API se denomina Promise, a diferencia de otras APIs, XMLHttpRequest o JQuery.ajax(), el objeto Promise no será rechazado con un estado de error del protocola HTTP, este se resolverá normalmente y será rechazado solo s la solicitud fue incompleta. Fetch por defecto no enviará o recibirá cookies.

La creación de un websocket en tiempo real sería de la siguiente manera.

1. Definimos nuestra variable constante API_URL, siendo “ruta”, la ruta del proyecto

```
const API_URL = window.location.protocol + "//" + window.location.host + "/ruta";
```

2. Inicializamos el API Fetch, siendo “hostname” la ip del websocket cliente

```
fetch(API_URL).then(function (response) {
    openWebsocketClient(hostname);
});
```

3. Definimos el protocolo, puerto y endpoint.

```
    var protocolo = "ws";
    var port = 7458;
    var endpoint = "/endpoint";
```

4. Inicializo la variable webSocketURL

```
webSocketURL = protocol + "://" + hostname + ":" + port + endpoint;
```

5. Creo el objeto WebSocket y abro cada una de sus funciones con las tareas a realizar.

```
function openWebsocketClient(hostname) {
    var protocolo = "ws";
    var port = 7458;
    var endpoint = "/endpoint";
    var webSocketURL = null;
    webSocketURL = protocol + "://" + hostname + ":" + port + endpoint;
    console.log("openWSConnection::Connecting to: " + webSocketURL);
    try {
        socket = new WebSocket(webSocketURL);
        socket.onopen = function (openEvent) {
            socket.send("Hi");

        };
        socket.onclose = function (event) {
            if (event.wasClean) {
                console.log('websocket cerrado);
            } else {
                console.log('websocket no cerrado');
            }
```

```
        };
        socket.onerror = function (event) {

            console.log("Error en la ejecución del websocket: " + e);

        };
        socket.onmessage = function (event) {
            let data = event.data;
            try {
                mostrartiemporeal(data);
            } catch (e) {
                console.error(e);
                data = null;
            }
            data = null;
        };
    } catch (exception) {
        console.error(exception);
    }
}
```

6. Cada opción representa una acción que se puede realizar.

    a. Onopen: Abre el puerto de comunicación y se envía una palabra, frase o valor definido para que websocket server acepte la conexión bilateral.

    b. Onclose: Cierra el puerto y deja de trasmitir, el evento finaliza.

    c. Onerror: Notifica un error en la ejecución del proceso.

    d. Onmessage: Recibe la información en tiempo real. Aquí realizamos las reglas de negocio.

7. Toda variable, al finalizar su funcionamiento debe culminar cambiándola a NULL

El proceso de visualización de datos sobre la venta de combustible se ejecuta desde el navegador, permitiendo que las acciones realizadas con el servidor tengan su prioridad. Con el pasar del tiempo no existe latencia visible y las acciones deseadas se ejecutan en el plazo previsto.

Tabla 28

Características del software en tiempo real aplicado a JavaScript con websocket y API Fetch

Característica	Resultado
Determinismo	- El sistema garantiza la ejecución de otras tareas al mismo tiempo. - El tiempo en ejecutar el procesamiento de datos sobre la venta es mínimo.
Responsividad	- La ejecución de la petición se procesa desde le navegador, por lo que no interfiere con los otros procesos que necesitan transaccionar desde el servidor. - El usuario al solicitar la ejecución de otras tareas no tiene efecto secundario en el procesamiento de los datos de los surtidores.
Usuarios controladores	- El procesamiento de los datos de ventas se ejecuta en segundo plano, permitiéndole al usuario priorizar la tarea deseada. - El proceso no tiene control sobre la memoria RAM a utilizar, pero lo usado por esta no afecta los otros procesos de la aplicación y/o servidor.
Confiabilidad	- Con el transcurso del tiempo la calidad del sistema no se degrada. - Si la trama recibida por el websocket está incompleta no afecta el funcionamiento de la aplicación, ni de los otros procesos, se actualiza lo recibido.
Operación a prueba de fallas duras	- Si el registro de un lado está incompleto no afecta en la visualización de los otros lados, y cuando recepta el siguiente registro lo actualiza. - El sistema es estable con el tiempo.

Para las pruebas de funcionamiento el sistema estuvo operando por 7 días en un terminal cliente zero, que se conecta con un acceso remoto y utiliza los recursos del servidor, obteniendo como resultado que el uso de memoria es mínimo y permite que otras tareas requeridas en el proceso se ejecuten con normalidad.

De las cinco características que una aplicación en tiempo real se cumple todas, con lo cual se verifica que la información mostrada de ventas está en Tiempo Real.

Llegando a este punto hay una pregunta que es: ¿Cuánto tiempo es tiempo real? Y la respuesta es que el tiempo es subjetivo, lo que para una persona es rápido para otra no, pero en software, para que una aplicación sea considera así debe cumplir características, ser capaz de producir el resultado desea en el tiempo esperado y que no tenga latencia e inestabilidad.

Teniendo la información procedo a identificar los distintos estados y los colores a utilizar en cada uno.

Tabla 29

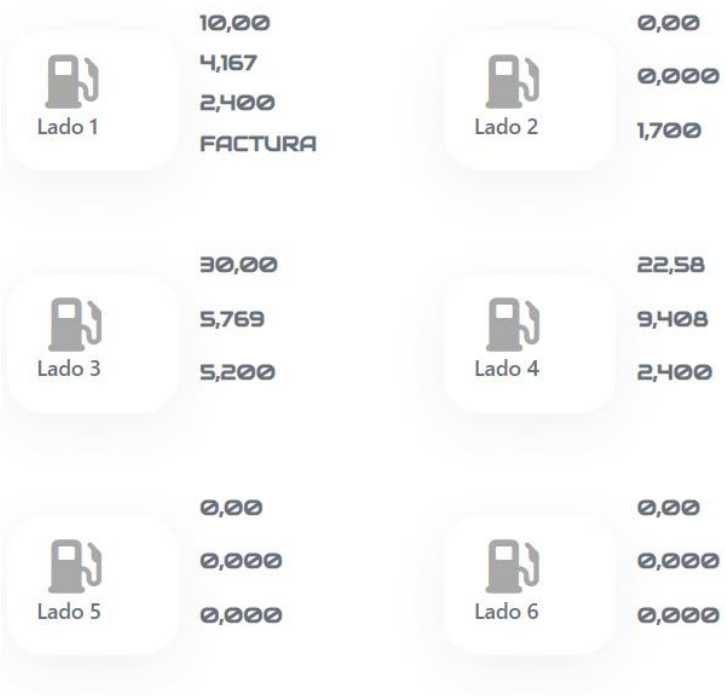
Asignación de colores a estado

Estado	Color	Color Hexadecimal	Descripción
Inactivo	Plomo	#A9A9A9	Indica que el lado no se encuentra en funcionamiento, por lo tanto no mostrará tampoco información de ventas.
Activo	Celeste	#14C8CD	Indica que el lado está listo para que el despachador venda combustible.
Despachando	Morado	#9E14CD	Indica que el lado está vendiendo, adicional del color se mostrará la venta en tiempo real.
Cerrado	Rojo	#CD1433	Indica que la venta terminó, la manguera cerró su flujo de combustible y espera que el despachador ingrese la información del cliente para pasar a estado activo.

La visualización sin colores definidos se la muestra en la siguiente imagen:

Figura 17

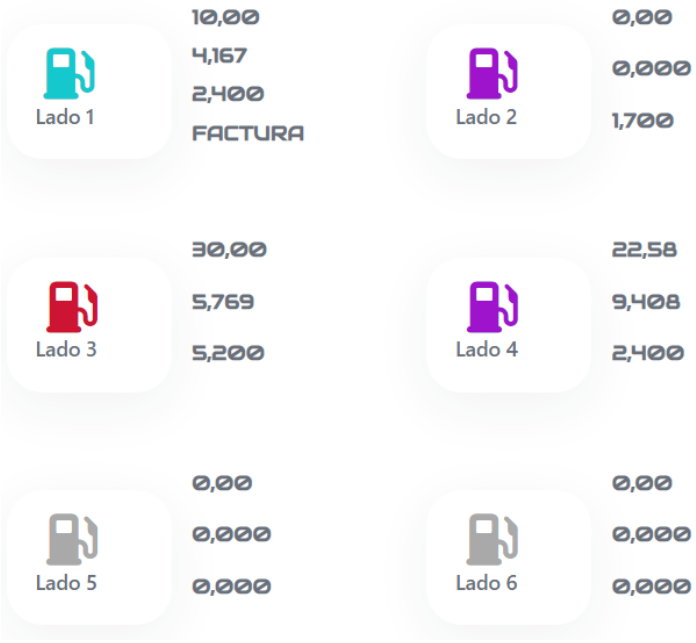
Vista de los lados sin colores



La visualización con colores definidos se la muestra en la siguiente imagen:

Figura 18

Vista de los lados con colores



La utilización de distintos colores para identificar los estados de cada venta, le ayuda al usuario a distinguir que acción realizar ya sea en el sistema o físicamente interactuando con el cliente. Por ejemplo:

- El lado 1 indica que está activo y no están despachando, está disponible para una siguiente venta.
- Los lados 2 y 4 indican que están vendiendo, se está surtiendo combustible al cliente.
- El lado 3 indica que su venta terminó, es momento de ingresar los datos de la venta y proceder a cobrar lo surtido.
- Los lados 5 y 6 están inactivos, el surtidor de gasolina puede estar apagado o dañado.

Sprint Backlog 2

Terminada la iteración 1 para la iteración 2 están contempladas las historias de usuario 4,5, y 6.

- Información del lado.

- Búsqueda de clientes registrados.

- Búsqueda de clientes nuevos y registro de placas.

Tabla 30

Sprint Backlog 2

ID	Responsable	Descripción de la historia de usuario	Tareas	Esfuerzo empleado en días
6	Juan José Pérez	Necesito que al presionar en el lado se visualice la información registrada.	Crear una vista para el ingreso de la información.	2
			Al presionar sobre verificar si existe información	1
7	Juan José Pérez	Necesito buscar la información registrada de un cliente	Crear buscadores por código, placa e identificación	2
			Crear un controlador para consultar los clientes	1
			Validar la información según la normativa del SRI.	1
			Mostrar la información en la vista	1
8	Juan José Pérez	Necesito buscar clientes no registrados	Validar la información de la identificación según el Registro Civil antes de consultar	2

			Crear una vista para la creación de clientes	1
			Crear un controlador de clientes nuevos	1
			Validar la información del cliente	1
		Necesito registrar placas	Crear una vista para registrar placas	1
			Crear un controlador para registrar placa	1
			Validar la placa a ingresar según el Servicio de Rentas Internas para la autorización de comprobantes electrónicos.	1

Sprint Review 2

Para este artefacto se requiere el verificar los resultados del Sprint Backlog 2, a continuación, se detalla su avance.

Tabla 31

Sprint Review 2

ID	Tarea	Iteración	Avance
6	Crear una vista para el ingreso de la información.	2	100%
	Al presionar sobre verificar si existe información.	2	100%
7	Crear buscadores por código, placa e identificación	2	100%
	Crear un controlador para consultar los clientes	2	100%
	Validar la información según la normativa del SRI.	2	100%
	Mostrar la información en la vista	2	100%
8	Validar la información de la identificación según el Registro Civil antes de consultar	2	100%
	Crear una vista para la creación de clientes	2	100%
	Crear un controlador de clientes nuevos	2	100%
	Validar la información del cliente	2	100%
	Crear una vista para registrar placas	2	100%
	Crear un controlador para registrar placa	2	100%
	Validar la placa a ingresar según el Servicio de Rentas Internas para la autorización de comprobantes electrónicos.	2	100%

Entregables Sprint 2

La iteración 2 tiene los siguientes entregables:

- Visualización de la información de cada lado.
- Búsqueda de clientes registrados.
- Búsqueda de clientes nuevos y registro de placas.

Visualización de la información

Para completar el proceso de venta, se tiene que registrar o actualizar la información en el lado que se encuentre surtiendo gasolina.

El despachador presiona sobre el surtidor del Lado deseado y se mostrará la ventana vacía o con información de tenerla. No existen mensajes de alerta.

Figura 19

Vista de Registro de Clientes sin información

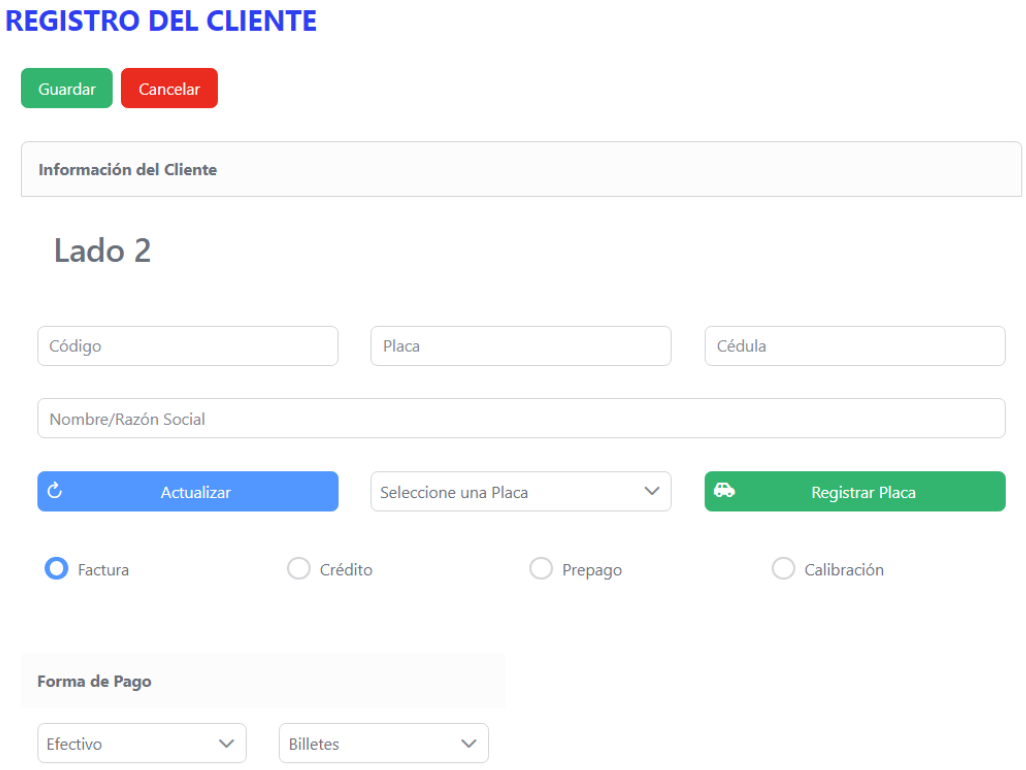
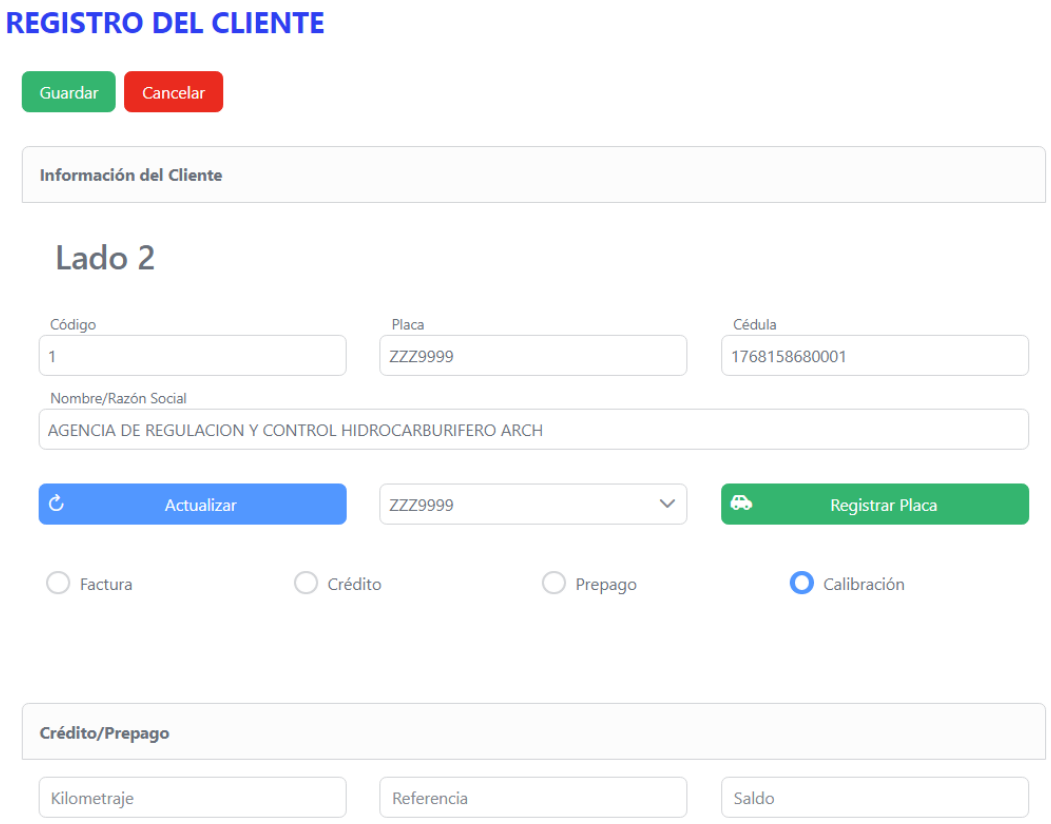


Figura 20

Vista de registro de clientes cuando existe información en la base de datos



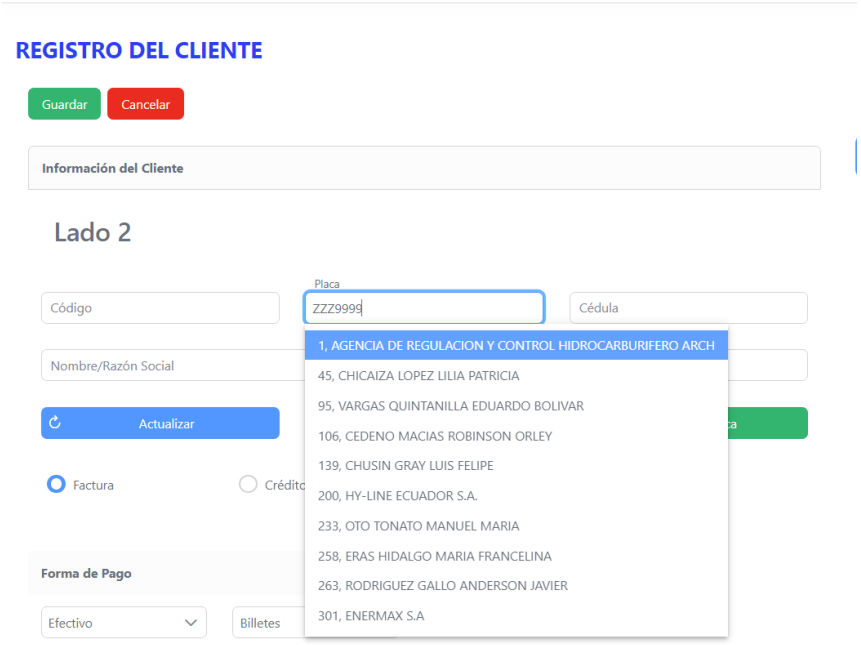
Búsqueda de clientes registrados

Para buscar un cliente registrado se lo puede hacer por código, placa o identificación, al encontrarlo el despachador valida la información y posteriormente se visualiza en pantalla. Adicional a la información del cliente, se registra el tipo de venta y la forma de pago.

1. Ingresar en el campo deseado lo que se desea buscar. A continuación, se mostrará una lista con las coincidencias favorables.

Figura 21

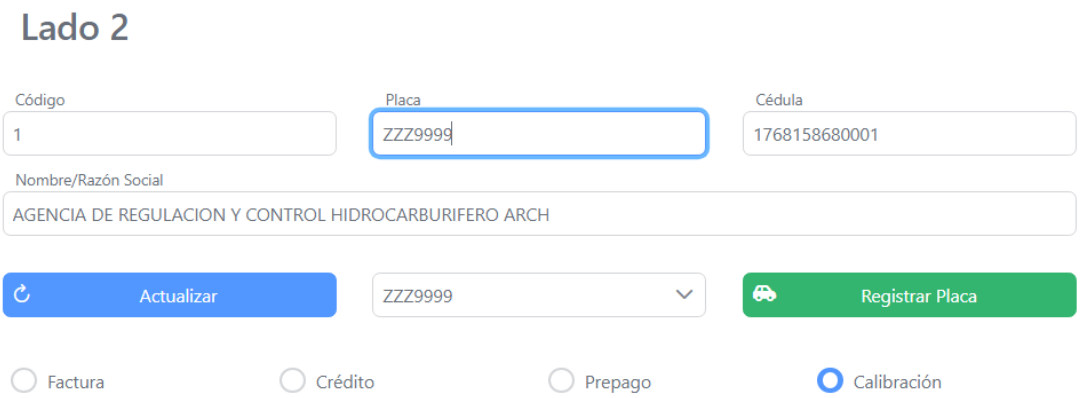
Búsqueda de un cliente registrado en la base de datos



2. El usuario selecciona el registro con el cliente correcto y se actualizan todos los campos de la vista.

Figura 22

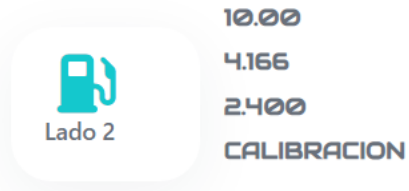
Selección de cliente registrado en la base de datos



3. Al presionar guardar registro la información y se actualiza la vista de los lados con el tipo de venta a realizar.

Figura 23

Vista del lado con información de cliente registrado



Búsqueda de clientes nuevos y registro de placas

Para el registro de clientes se tiene que validar la información de identificación con el algoritmo del registro civil para cédulas y para R.U.C la normativa indica que deben ser 13 caracteres con 001 al final. Para la explicación del algoritmo detallo el siguiente ejemplo con la identificación 1804474466:

1. El número consta de 10 dígitos.
2. Los dos primeros dígitos son el código de la provincia: 18, como está dentro del rango 01 a 24 es correcto.
3. El tercer dígito es 0, lo cual indica que es válido.
4. El número verificador es 6, es el último dígito de la identificación.
    a. Se agrupan todos los dígitos menos el verificador en pares e impares.

Tabla 32

Agrupación de dígitos

	Uno	**Dos**	**Tres**	**Cuatro**	**Cinco**	**Seis**	**Siete**	**Ocho**	**Nueve**
Dígitos	1	8	0	4	4	7	4	4	6
Impares	1		0		4		4		6
Pares		8		4		7		4	

b. A los impares se los multiplica por dos, y si el resultado es mayor a nueve, se le resta nueve.

Tabla 33

Multiplicación de valores impares

Impares	1x2		0z2		4x2		4x2		6x2
	2		0		8		8		12-9
									3

c. Se suma los dígitos pares más los del resultado anterior.

Tabla 34

Sumatoria de dígitos

Impares	2	+	0	+	8	+	8	+	3	=21
Pares		8	+	4	+	7	+	4		=23
Suma total										**44**

d. Se obtiene el módulo del valor anterior.

44%10=4

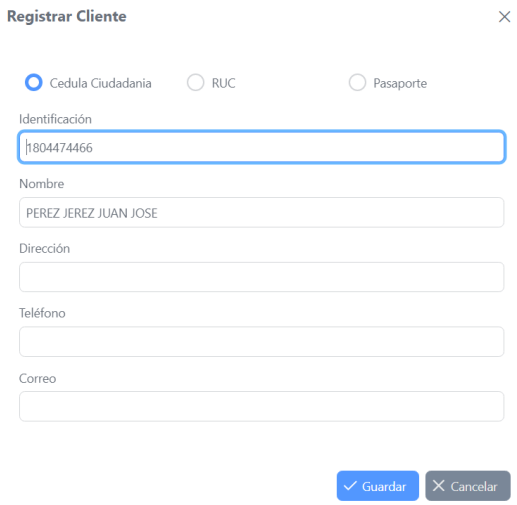
e. Al número 10 se le resta el resultado del anterior paso.

10-4=6

Luego de validar la información se procede a buscar en un Servicio Web Rest de Gasintec S.A., este retornará la información del cliente siendo el nombre el campo obligatorio y opcionales dirección, teléfono y correo electrónico.

Figura 24

Registro de cliente nuevo



El despachador procede a ingresar la información restante y presiona guardar para registrar el nuevo cliente.

En el caso del registro de placas la validación la da la Agencia Nacional de Tránsito y el Servicio de Rentas Internas y puede ser las siguientes:

- **Vehículo:** Tiene un tamaño de siete caracteres, tres letras, cuatro números.
- **Moto:** Tiene un tamaño de seis caracteres, dos letras, tres números, una letra.
- **Cuantía Doméstica:** Tiene un tamaño de ocho caracteres, inicia con CU y finaliza con seis dígitos de autorización otorgada para la cuantía.

- **Maquinaria Agrícola:** Tiene nueve caracteres, inicia con MAQN y finaliza con cinco dígitos de la placa o autorización otorgada por el Ministerio de Obras Públicas u otra entidad de control.
- **Cuerpo Diplomático:** Tiene un tamaño de seis caracteres, inicia con CD y finaliza con cuatro números.
- **Organismo Internacional:** Tiene un tamaño de seis caracteres, inicia con OI seguido de cuatro números.

Para la validación se utiliza REGEX, una vez solicitado el registro de una placa nueva, el valor entra en un bucle hasta que el valor coincida con algún patrón. Si no coincide se mostrará un mensaje de error.

Figura 25

Vista de registro de placas

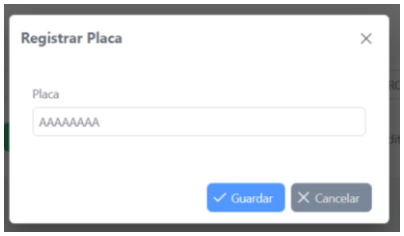


Figura 26

Mensaje de error cuando la placa no coincide con ningún patrón de validación

Capítulo V: Deployment

Para el proceso de Deployment se utilizó el servidor web Wildfly en su versión 13. Wildfly es un servidor liviano y potente de código abierto basado en JBOSS. Se lo utiliza por su rápida configuración y disponibilidad para diferentes sistemas operativos. El archivo de despliegue tiene extensión ear.

Configuración

La configuración será para sistema operativo Windows.

1. Configuración de la conexión a la base de datos.

La base de datos utilizada es PostgreSQL 13, y la librería a utilizar es la postgresql-42.2.5.jar. La librería se copia en la dirección de la figura 27 y se crear el archivo module.xml

Figura 27

Ruta de librería en el servidor Wildfly

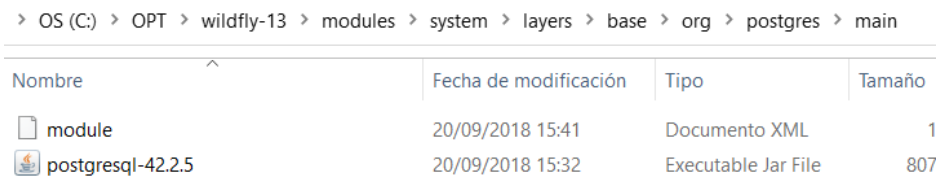


Figura 28

Archivo module.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.3" name="org.postgres">
<resources>
        <resource-root path="postgresql-42.2.5.jar"/>
    </resources>
    <dependencies>
        <module name="javax.api"/>
        <module name="javax.transaction.api"/>
        <module name="javax.servlet.api" optional="true"/>
    </dependencies>
</module>
```

2. Configuración de uso de memoria RAM

La memoria RAM a utilizar depende de la capacidad del equipo, no se puede utilizar la totalidad porque existen otras aplicaciones que también lo requieren y no se puede utilizar el valor por defecto (64MB) porque la aplicación no funcionará correctamente. Se utiliza el 25% si la memoria RAM tiene una capacidad de 8GB o menos y puede llegar al 50% en valores superiores. Para configurar la Java Virtual Machine (JVM) utilizaremos el siguiente comando:

Figura 29

Comando de configuración JVM

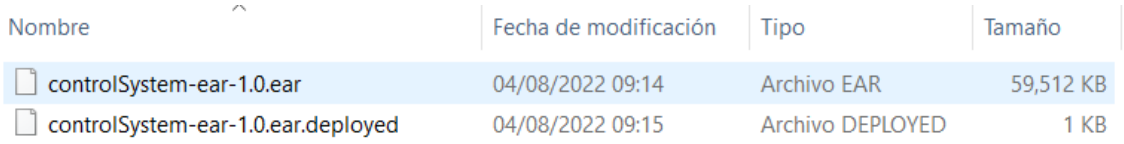
```
set "JAVA_OPTS=-Xms4096M -Xmx8192M -XX:MetaspaceSize=4096M -XX:MaxMetaspaceSize=4096m"
```

3. Despliegue

El archivo generado para la aplicación tiene extensión ear, para ser desplegado, se lo agrega al servidor y se desplegará un archivo con extensión deployment si fue correcto, caso con extensión failed.

Figura 30

Archivos de deployment



Verificación

Desplegada la aplicación se verifica la funcionalidad desde el navegador colocando la ruta de acceso.

Figura 31

Aplicación no desplegada

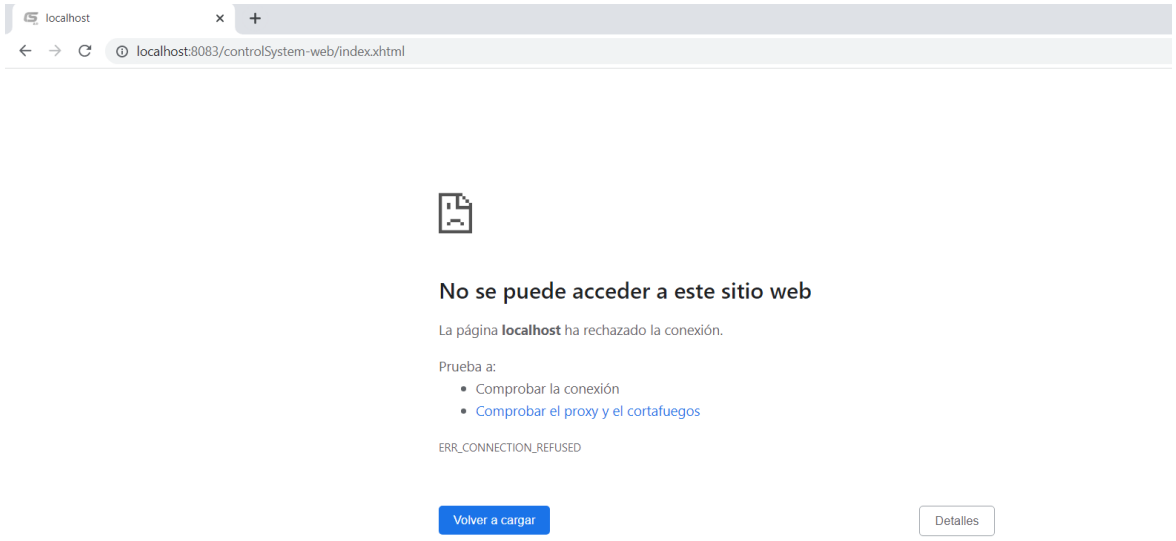
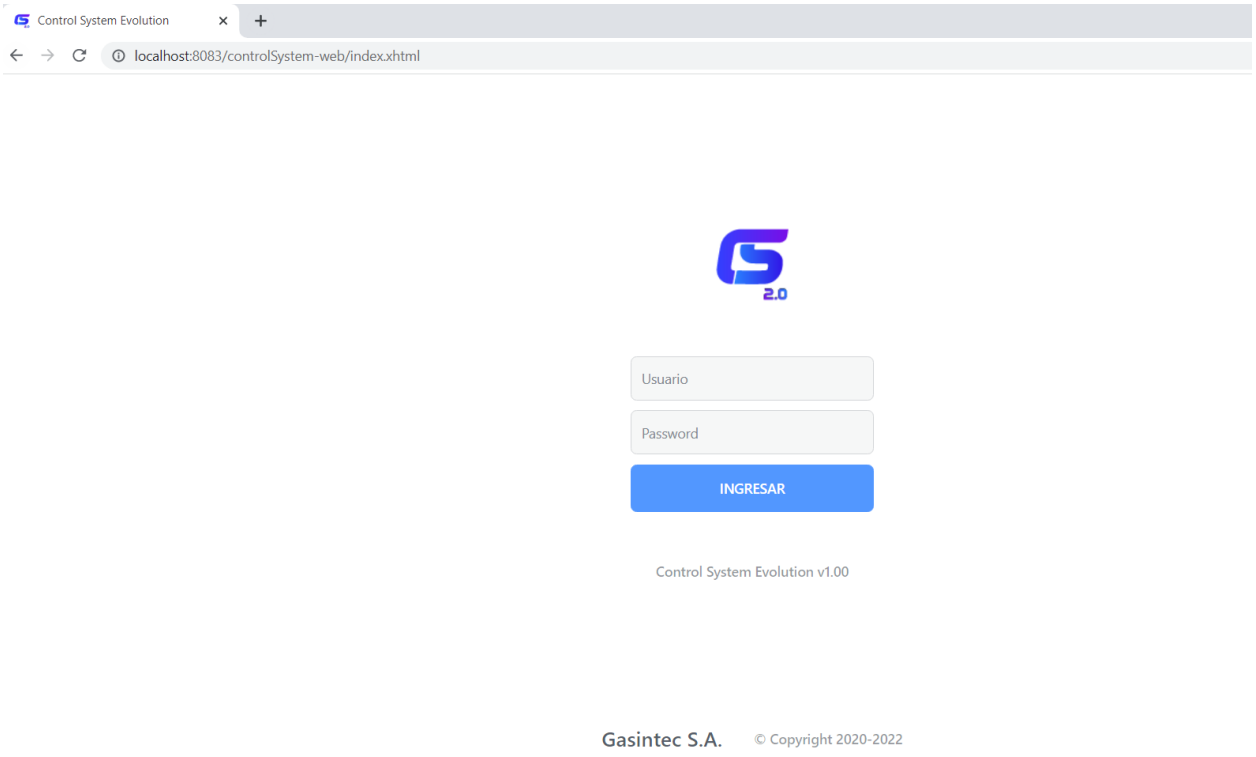


Figura 32

Aplicación desplegada



Capítulo VI: Conclusiones y Recomendaciones

Conclusiones

- El protocolo websocket con el API Fetch de JavaScript facilita la visualización de la información en tiempo real y optimiza los recursos de hardware del servidor. Minimiza la latencia y permite que el usuario priorice las acciones a ejecutar por la aplicación.

- No se puede definir cuanto tiempo es tiempo real, puesto que depende de la percepción del usuario, el software en tiempo real se basa en la aplicación de todas sus características sin excepción.

- El software en tiempo real facilita al despachador de una gasolinera el control sobre el combustible surtidor y valor vendido en efectivo, así como conocer el estado de su venta mediante una respuesta gráfica de la pantalla.

- WebSocket es mejor para el manejo de tiempo real que Poll de Primefaces, Websocket abre un túnel de comunicación bidireccional segura en el cual el cliente no necesita realizar una petición para obtener información, mientras que Poll depende del intervalo de tiempo ingresado para su ejecución.

Recomendaciones

- La optimización de recursos es fundamental para el funcionamiento correcto de un sistema en tiempo real, el tener en pruebas un hardware potente puede dar una falsa ilusión de que se llegó al objetivo, se recomienda que antes de salir a producción se realice pruebas en el hardware mínimo que puede tener.

- Existen varias APIs de JavaScript para el manejo del tiempo real, y constantemente hay nuevas librerías, APIs que facilitan su ejecución. Antes de empezar el proyecto se recomienda revisar cuál sería la más óptima según lo deseado.

- Recomiendo la revisión bibliográfica actualizada, las inquietudes de desarrollo o entendimiento de un problema, alguien ya lo puede haber resuelto.

- Recomiendo analizar el usuario principal del sistema a realizar para la creación de las diferentes vistas y facilitar su uso. Por ejemplo, el sistema es vendido al dueño de una gasolinera, pero él no es el usuario del sistema, el que está en contacto siempre es el despachador, en algunos casos adultos mayores, o personas con un mínimo de estudio. un software no amigable es posible que lo desechen.

Bibliografía

Burns, W. (2009). *Real-time systems and programming languages.* Addison-Wesley.

Comercio, E. (26 de Junio de 2012). *Breve reseña sobre la historia petrolera del Ecuador*. Obtenido de El Comercio: https://www.elcomercio.com/actualidad/negocios/breve-resena-historia-petrolera-del.html

Cuenca Navarrete, L. G. (2019). ESTUDIO DEL PROCESO DE DESPACHO DE COMBUSTIBLES Y LA CALIDAD DEL SERVICIO EN LA GASOLINERA CORPORACIÓN CHICAIZA DEL CANTÓN SALCEDO. *Ambato: Universidad Tecnológica Indoamérica*, 40.

Demo, J. P., Painefilu, J. P., Ferreira Szpiniak, A., & Zorzán, F. A. (2014). Chat, Pizarra Virtual, Aulas Modulares Virtuales. *IX Congreso de Tecnología en Educación & Educaci´on en Tecnología*, 13.

Evequoz, O. S. (2005). Pérdidas evaporativas por almacenamiento y distribución de combustibles en estaciones de servicio. Análisis de su problemática y propuesta de marco regulatorio local. *Universidad Nacional de Córdoba - Argentina*, 25.

Fette, I. &. (2011, Septiembre). *The websocket protocol.* Retrieved from Obtenido de The websocket protocol: https://tools.ietf.org/html/draft-ietf-hybi-thewebsocketprotocol-17

Gasintec. (2020). *Quienes Somos*. Retrieved from Gasintec: https://gasintec.com/quienes-somos/

Gupta, Y. D. (2020). Real-time monitoring using AJAX and WebSockets. *Journal of Statistics and Management Systems*, 125-134.

MARTEL, A. (2015). Gestión práctica de proyectos con Scrum: Desarrollo de software ágil para el Scrum Master. *Amazon*.

MDN. (2022, 05 23). *Uso de Fetch*. Retrieved from MDN web docs: https://developer.mozilla.org/es/docs/Web/API/Fetch_API/Using_Fetch

Mendoza, J. G. (2017). ANÁLISIS DEL USO DE WEBSOCKETS PARA IMPLEMENTAR APLICACIONES WEB EN TIEMPO REAL. *Pistas Educativas*, 3-.

Murley, P. M. (2021). Websocket adoption and the landscape of the real-time web. *Proceedings of the Web Conference 2021*.

Operativos, B. S. (2017, Abril 27). *Características de los sistemas de tiempo real*. Retrieved from Bitáctora Sistemas Operativos: https://chsos20171914562blog.wordpress.com/2017/04/27/caracteristicas-de-los-sistemas-de-tiempo-real/

Pérez, J. E. (2019). *Introducción a JavaScript.* www.librosweb.es.

Primefaces. (2022). *Primefaces Documentation*. Retrieved from Primefaces: https://primefaces.github.io/primefaces/12_0_0/#/?id=main

Ramirez Claros, D. E. (2020). Comunicación en tiempo real por medio de websockets para el control del prototipo Remington. *Remington. Memorias.*

Trigás Gallego, M. (2012). Metodologia scrum.

WebSocket.org. (2015). *This is websocket.org*. Retrieved from This is websocket.org: http://websocket.org/aboutwebsocket.html